A graphical analysis tool for hardware design

Mariusz Pawel Pytel
s030166

Kongens Lyngby 2007
IMM-M.Sc-2007-111

Technical University of Denmark
Informatics and Mathematical Modeling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-MSc:2007-111

Abstract

In order to meet requirements imposed by a rapid grow in complexity of chips, existing design and verification methodologies have to be extended and a new ones created.

The main objective of this master thesis is to investigate possibilities for improvement in creation and verification of high-level hardware description language models. GEZEL is an example of such a high-level hardware description language, which models hardware according to the semantics of the finite-state machine with datapath (FSMD). This feature of the language allows use of symbolic model checkers like NuSMV in the automated validation process. Both, GEZEL and NuSMV are presented in the report, and their parts relevant to the project explained.

As a part of the project a graphical tool was developed that visualizes hardware models and supports hardware architect in development process. The tool is able to check, whether model obey well-formedness conditions and provides means for full functional check by generation of NuSMV representation of the model, which further can be validated using this symbolic model checker.

Keywords: hardware descriptions, graphical representation, model-checking, verification, GEZEL, NuSMV.

Preface

This thesis was prepared at Informatics and Mathematical Modeling (IMM) department at the Technical University of Denmark (DTU) in partial fulfillment of the requirements for acquiring the M.Sc. degree in Computer Science and Engineering. The project has been made during the period 14th June to 31st December under supervision of Michael R. Hansen and Jan Madsen, and corresponds to 30 ECTS-points.

……..
Mariusz Pawel Pytel

Lyngby, December 2007

Acknowledgements

I would like to thank my supervisors Michael R. Hansen and Jan Madsen for their help and support during the project. I especially appreciate many constructive discussions we had during our weekly meetings.

Additional thanks go to Martin Fränzle (University of Oldenburg, Germany) for his expertise on NuSMV, and Aske Brekling for his valuable suggestions concerning graphical representation and verification of GEZEL programs.

Finally, I would like to thank my parents Hanna and Marian, and my girlfriend Joanna Balukiewicz for supporting me through the last six months.

Table of Contents

1 Introduction

1.1 Introduction to the hardware architectures

Since the invention of semiconductor technology in 1958, its development has been rapid. Decrease in the size of transistors caused increase in the number of them being placed on a single chip. This results in grow of chip complexity, which is a product of chip size and its gate density. Number of transistors doubles, roughly, every 18 months, following the Moor's law - prediction made by co-founder of Intel Gordon E. Moore in 1965. Figure 1.1 shows change in transistor count placed on a single chip over the last 40 years. According to Intel and IBM researchers, it is expected that at least for next 10 years or perhaps even longer, this trend should be sustained. Current state-of-the-art processors, like Dual-Core Intel Itanium 2, have over 1.000 million transistors on a single chip.

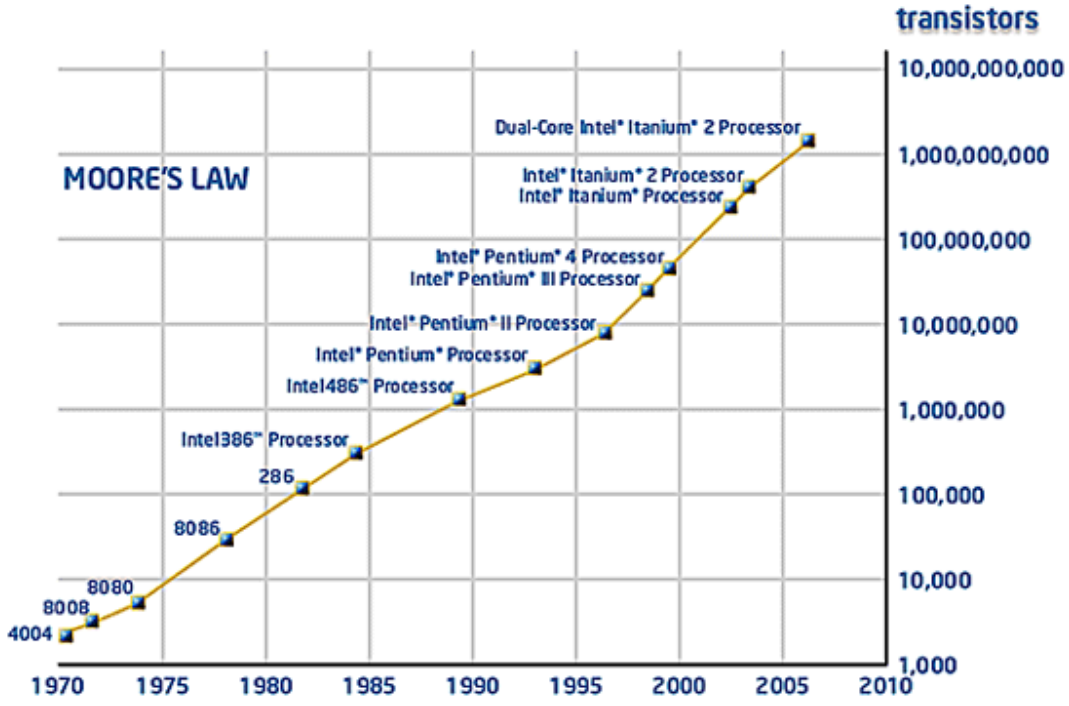

Figure 1.1: Moore's law – grow of transistor counts for processors. [7]

Nowadays, integrated circuit based systems are ubiquitous. Its influence on our everyday life is enormous. Computers are present everywhere in our surroundings. Except obvious

examples like personal computers and laptops, we use them, embedded in products such as: mobile phones, cars, TVs, stereos, clocks, ovens, refrigerators and many others. Typical mobile phone contains units for digital signal processing and control, hardware accelerating specific domains like graphics or cryptography. Car contains over 30 processors for example to control airbags, air conditioning, ABS, fuel injection and control panel. Predictions state that in the close future computers will be even more present. Pervasive computing models share a vision of small, inexpensive processing devices distributed all over human environment, being embedded in our clothes and houses, present everywhere in a form of intelligent dust, even implanted in our bodies, communicating with each other, controlling and measuring. As we can see, digital systems play important role in today's life. Being strategic materials for modern innovative products and services, they are involved virtually in all humans' activities and have a great impact on the overall performance of our society.

Together with new possibilities resulted from this extremely rapid progress in semiconductor technology, new difficulties arise. All those digital systems have to be designed and validated at the lowest possible cost and under tight time circumstances, since even the shortest delay, in such a dynamic industry, may result in the complete failure of the product. Growing complexity of the systems makes designers unable to fully utilize all possibilities provided by microelectronic technology. Need for application-specific systems with low or medium production volumes, as well as high quality requirements imposed on the systems embedded in military, space, avionic or medical equipment results in a fact that complexity and quality of the systems, its production time and cost are limited by the methodologies and tools used in design, rather than by technology itself. Due to the increase in complexity, new methods and tools have to be provided to the hardware architects, as the classic design methods, with split hardware software design path, are no longer sufficient and any substantial improvement in this field may be only achieved through development of those new methodologies.

In recent years top-down approaches where consequently adopted and led designer from logic and transistor-level design to abstract programming. Today's standard for hardware design is use of VHDL and Verilog. Both languages support high-level architectural description, but still are capable of incorporation of low-level details for optimization purposes. It allows productions of highly optimized systems, but ties hardware description to specific technology. GEZEL is an example of language for description of micro-architectures that may be considered at the higher level of abstraction then Verilog and VHDL. It has no explicit constructs for clock signal, which makes GEZEL independent on any particular implementation technology. Language syntax consists of elements with a clear hardware meaning: signals, registers, controllers, etc. Figures 1.2 and 1.3 show implementation of GCD algorithm, written in, respectively, GEZEL and pseudo code.

```
dp euclid(in  m_in, n_in : ns(16);
          out gcd        : ns(16)) {
  reg m, n               : ns(16);
  reg done               : ns(1);
  reg factor             : ns(16);

  sfg init    { m = m_in; n = n_in; factor = 0; done = 0; gcd = 0;
                $display("cycle=", $cycle, " m=", m_in, " n=", n_in); }
  sfg shiftm  { m = m >> 1; }
  sfg shiftn  { n = n >> 1; }
  sfg reduce  { m = (m >= n) ? m - n : m;
                n = (n >  m) ? n - m : n; }
  sfg shiftf  { factor = factor + 1; }
  sfg outidle { gcd = 0; done = ((m == 0) | (n == 0)); }
  sfg complete{ gcd = ((m > n) ? m : n) << factor;
                $display("cycle=", $cycle, " gcd=", gcd); }
}

fsm euclid_ctl(euclid) {
  initial s0;
  state s1, s2;

  @s0 (init) -> s1;
  @s1 if (done)                then (complete)          -> s2;
      else if ( m[0] &  n[0]) then (reduce, outidle)   -> s1;
      else if ( m[0] & ~n[0]) then (shiftn, outidle)   -> s1;
      else if (~m[0] &  n[0]) then (shiftm, outidle)   -> s1;
                              else (shiftn, shiftm,
                                    shiftf, outidle)   -> s1;
  @s2 (outidle) -> s2;
}
```

Figure 1.2: GEZEL code for greatest common divisor algorithm.

```
gcd(a,b)
{
   while (a > 0 && b > 0)
   {
      if (!isEven(a) && !isEven(b))
      {
         a = a > = b ? a - b : b;
         b = b > a ? b - a : a;
      }
      else if (isEven(a) && !isEven(b))
         a = a / 2;
      else if (!isEven(a) && isEven(b))
         b = b / 2;
      else
      {
         a = a / 2;
         b = b / 2;
      }
   }
   retrun a > b ? a : b;
}
```

Figure 1.3: Pseudo code for GCD algorithm presented on the figure 1.2.

In a GEZEL program there is a clear distinction between what is data processing and what is control. Constructs responsible for both functions may be seen on the figure 1.2, where are defined as respectively *dp euclid* and *fsm euclid_ctl.* Data processing part is called a datapath and it consists of finite set of actions, while control part is called a controller and its task is to schedule execution of those actions from the datapath. An action, called in GEZEL language signal flow graph, is a set of operations on the datapath variables and ports. In the figure 1.2 examples of signal flow graphs are: *sfg init, sfg shiftm, sfg shftn, sfg reduce.*

```
D:\gezel-2.2-win\build\bin>fdlsim -d euclid.smv 5
> Cycle 1
*** INFO: Cycle 1: RTCTL
euclid_ctl: euclid_ctl.s0 -> euclid_ctl.s1
*** INFO: Cycle 1: eval_out ip
*** INFO: Cycle 1: eval
*** INFO: Cycle 1: eval ip
*** INFO: Cycle 1: disp
cycle=0 m=6 n=3
                    0                    6 euclid.m
                    0                    3 euclid.n
> Cycle 2
*** INFO: Cycle 2: RTCTL
euclid_ctl: euclid_ctl.s1 -> euclid_ctl.s1
*** INFO: Cycle 2: eval_out ip
*** INFO: Cycle 2: eval
*** INFO: Cycle 2: eval ip
*** INFO: Cycle 2: disp
                    6                    3 euclid.m
> Cycle 3
*** INFO: Cycle 3: RTCTL
euclid_ctl: euclid_ctl.s1 -> euclid_ctl.s1
*** INFO: Cycle 3: eval_out ip
*** INFO: Cycle 3: eval
*** INFO: Cycle 3: eval ip
*** INFO: Cycle 3: disp
                    3                    0 euclid.m
> Cycle 4
*** INFO: Cycle 4: RTCTL
euclid_ctl: euclid_ctl.s1 -> euclid_ctl.s1
*** INFO: Cycle 4: eval_out ip
*** INFO: Cycle 4: eval
*** INFO: Cycle 4: eval ip
*** INFO: Cycle 4: disp
                    0                    1 euclid.done
> Cycle 5
*** INFO: Cycle 5: RTCTL
euclid_ctl: euclid_ctl.s1 -> euclid_ctl.s2
*** INFO: Cycle 5: eval_out ip
*** INFO: Cycle 5: eval
*** INFO: Cycle 5: eval ip
*** INFO: Cycle 5: disp
cycle=4 gcd=3
```

Figure 1.4: GEZEL simulation output for GCD algorithm.

GEZEL environment provides means for simulation and translation into synthesizable VHDL code. An example of such a simulation output for the model presented in the figure 1.2 may be seen in the figure 1.4.

The GEZEL environment does not provide any means for verification of created systems. This thesis tries to address the issue. As a part of the project, a tool capable to model system behavior in terms of symbolic model checker NuSMV was developed. This allows performing formal verification of GEZEL specifications. In addition, the tool presents graphical representations of a GEZEL model to the user, and supports its edition and modification in a user-friendly, interactive environment.

1.2 Related work

Current standard for hardware design are three common hardware description languages: VHDL, Verilog and SystemC. Each of those languages can be considered on the lower abstraction level in comparison to GEZEL.

VHDL (Very High Speed Integrated Circuits Hardware Description Language) was developed on a request from the American Department of Defense. The language was supposed to describe hardware in a way which is readable both for humans and machines and force developer to write code in a structured way, so the source code can serve as a kind of specification document itself. The most important concept was to cope with parallelism of digital systems. VHDL is mainly used for the development of Application Specific Integrated Cicuits (ASICs). Tools for the automatic transformation of VHDL code into a physical hardware are available. This process of transformation is called synthesis and is an integral part of current design flows.

Verilog was invented by Phil Moorby and Prabhu Goel at Automated Integrated Design Systems in 1983. The language supports the design, verification, and implementation of analog, digital, and mixed-signal circuits at various levels of abstraction. In order to make the language familiar to engineers and instantly accepted, its syntax was based on C programming language. Control flow keywords, printing routines, operators and precedence are all similar to that language. The main difference from a conventional programming language is the execution of statements that is not strictly sequential. A subset of the language is synthesizable. If the hardware description consists only of synthesizable elements, it can be translated into hardware.

SystemC was developed by Synopsys Inc. for system simulation and synthesis. It is a set of library routines and macros implemented in C++. It makes simulation of concurrent process possible. Processes are described by the C++ syntax. They can communicate in a simulated real-time environment, using signals of data types provided by C++, additional ones provided by SystemC and defined by user. SystemC is both a hardware description language and a simulation kernel. The code compiles together with the library simulation kernel, resulting in executable, which behaves like a described model when it is run.

System validation is one of the most important activities in the design of hardware architectures process. Debugging consumes more than 70% of development time. Generally, verification may be divided into two kinds: simulation-based approaches and formal approaches. Simulation based approach can provide system architect with information about general behavior of the system. It is a prevalent approach due its simplicity and close connection to the design activities, however it cannot ensure correctness of the whole model, since it deals only with the portions of the model that are put under investigation, by a specific input vectors - not all traces are checked. Moreover, generation of input vectors is itself error-prone and time consuming process. On the other hand, formal verifications which address those drawbacks of simulation-based approaches, by checking all traces, need to deal with complexity of the system that is being verified. Usually, this problem is solved by creation of an abstract model of the system and performing verification on this approximated model. In [2] UPPAAL, a model-checking tool based on timed automata, is proposed as a back-end for formal verification of GEZEL programs. Translation of GEZEL models into terms of UPPAAL is specified, which allowed verification of several hardware specifications, for example Simplified DES and Euclidean algorithms.

1.3 Thesis objective

This master thesis is an attempt to investigate area for possible improvement in design and validation of hardware architectures. Main objectives may be specified as follows:

- Create graphical representation for hardware architecture models created by GEZEL-like languages.

- Design interactive tool for creation and edition of such a models.

- Develop formal verification strategy for those models.

1.4 Structure of the thesis

The thesis is structured as follows:

- Chapter 2 gives general introduction to the GEZEL language. Main concepts are explained and the environment for the simulation presented.

- Chapter 3 presents a graphical tool for modeling and analysis of hardware architectures developed as a part of this project.

- Chapter 4 discusses verification strategy for hardware models, by the use of an external symbolic model checker - NuSMV. Input language for NuSMV is introduced and translation of the hardware models, created using the tool presented in the chapter 3, to NuSMV explained.

- Chapter 5 explains ideas behind design and implementation of the graphical tool. The structure, main components and algorithms are all presented in this chapter.

2 The GEZEL language

Modern digital designs use specialized, but still programmable components, including traditional processors, FPGA, ASIP, DSP and many others, to meet their performance and energy efficiency constraints. Functions which traditionally were limited to software now are accelerated by the use of hardware, which is created concurrently to software. GEZEL is an environment targeted toward this kind of designs. It is an open environment for exploration and simulation of multiprocessor systems on chip and embedded hardware. The hardware is captured in a deterministic, cycle-true and implementation oriented language, which is a part of this environment.

GEZEL language allows implementation of the micro-architecture of domain-specific processors. It uses cycle true semantics and models hardware according to the semantic of finite state machine with datapath (FSMD). The simulation environment is scripted for fast edit-load-simulate cycles, in contrast to compiled hardware models, where every change requires long recompiling. Simulation of hardware provided by GEZEL is scripted, but still, effective, for cycle-true simulations achieved performance is comparable with typical compiled-code environments. Back end for the simulation is an open C++ library, which enables to integrate GEZEL into different host environments. There are co-simulation interfaces available to several instruction-set simulators, for example ARM, Leon, SH-DSP, and also to SystemC and Java. By the use of third party instruction-set simulators together with GEZEL, a programming platform can be constructed for co-processor and multiprocessor design, while standalone it works as a hardware exploration environment. Level of abstraction in GEZEL design may be raised by the use of library blocks, pre-compiled, user supplied blocks written in C++ provided as dynamic libraries, that run much faster, then cycle-true models, and at the outside look like regular GEZEL datapaths. Those library blocks, provide an excellent basis for simulation of intellectual property models and make it possible to implement functions that otherwise could not be created using GEZEL code, such as special types of IO or host system function calls. After validating GEZEL models can be easily translated into VHDL for hardware synthesis as well as to C++. Extra support for stimuli capture is present in GEZEL, allowing simulations to be reconstructed on the hardware models.

2.1 FSMD

GEZEL model is created by composition of basic building blocks. Each of those blocks, models hardware according to the semantic of the Finite State Machine with Datapath (FSMD), presented on the figure 2.1.

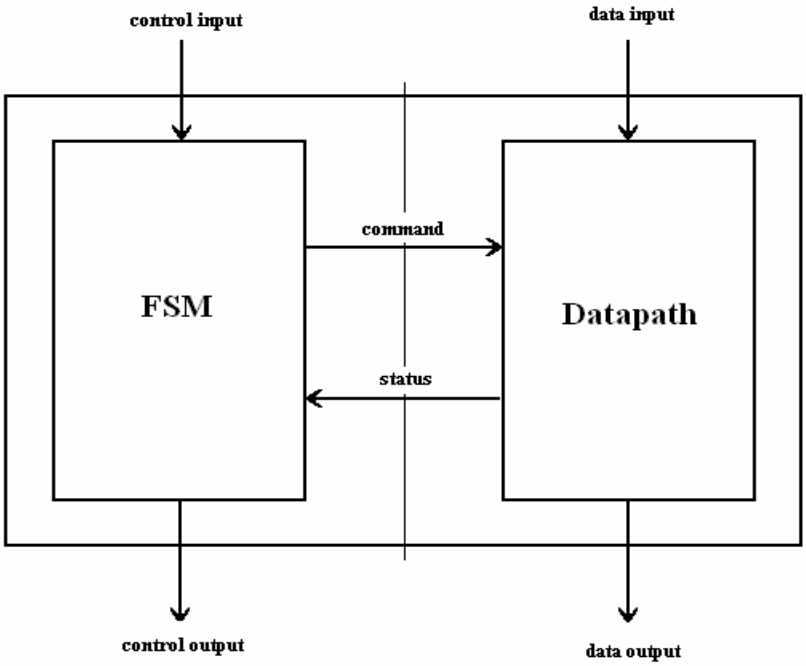

Figure 2.1: Model of finite state machine with datapath.

FSMD model expresses both datapath operations as well as control operations. However it makes a clear distinction between what is control and what is data processing. FSMD consists of two cross-coupled state machines. One plays the role of the controller, the other plays the role of the datapath. Controller holds values necessary to determine next control step, while datapath is responsible for storage of values needed for expression evaluation. FSMD model is presented on the figure 2.1.

A datapath consists of local variables - signals and registers, input and output ports and set of signal flow graphs – named actions on datapath variables. A controller is expressed in a form of finite state machine that executes in a deterministic way a set of actions from datapath. Decision which actions will be executed during specific cycle is based on the current state of the controller and values of the datapath registers. Figure 2.2 presents definition scheme for controller and datapath in GEZEL language.

Datapath

```
dp dp_name (port list) {
  register and signal declarations
  sfg name_1 { actions }
  sfg name_2 { actions }
  ...
}
```

Controller

```
fsm controller_name (dp_name) {
  state declaration
  @state_1 transition_1
  @state_2 transition_2
  ...
}
```

Composition

```
system id {
  dp_name_1(sig_1_1, sig_1_2, ... )
  dp_name_2(sig_2_1, sig_2_2, ... )
  ...
}
```

Figure 2.2: Pattern for a GEZEL program.

All components that compose GEZEL system are always active and operate in parallel. Execution of components is synchronized. During every cycle, for each component c_i, there is exactly one transition of controller $ctrl_i$, that result in execution of actions $act_{i,k}$, … , $act_{i,l,}$. Flow of the GEZEL program may be seen on the figure 2.3.

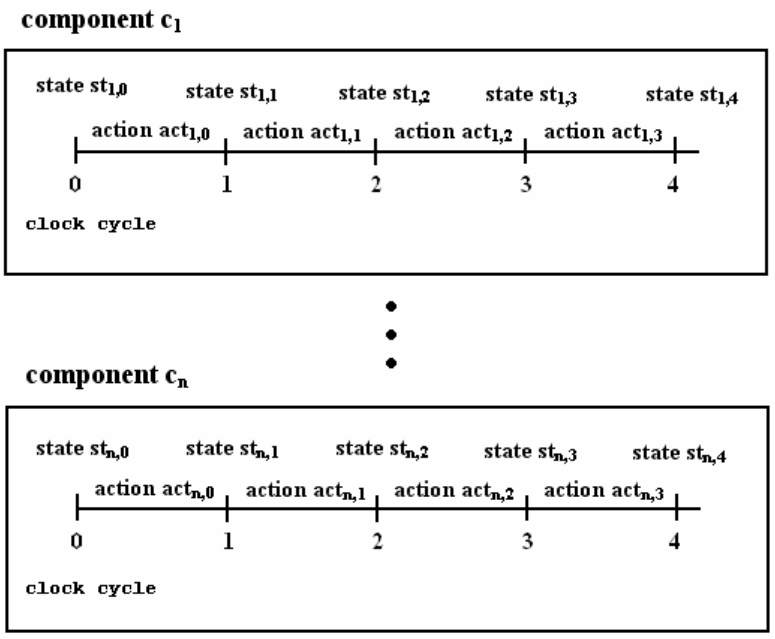

Figure 2.3: Flow of a GEZEL program.

Steps for a complete clock cycle in FSMD may be viewed on the figure 2.4.

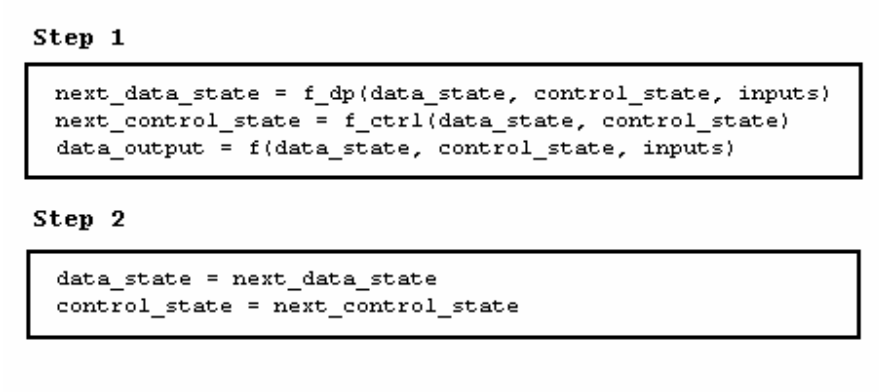

Figure 2.4: Complete cycle for GEZEL

At the beginning of each cycle output values and next states, both for controller and datapath are calculated. Next state for datapath is evaluated based on its inputs, registers and state of the controller, while for controller calculations are performed on the basis of its own state and the state of datapath registers. Finally, at the end of the cycle, in order to prepare for the next one, obtained next state values are copied into state values.

As we can see both state machines are cross-coupled. Information exchange between the two includes:

a) Conditions - reading state of datapath registers from controller in order to determine next control step.
b) Instructions - executing set of operations on datapath elements based on the state of controller.

FSMD provides separate modeling technique for data processing and for control processing. Datapaths are modeled with expressions on signals and registers, while controllers are modeled with state transition graphs. The logic implementation style of the two also shows differences. A datapath with operators typically exhibits a regular logic style. A controller on the other hand exhibits an irregular logic style. Examples of both styles can be seen on the figure 2.2.

2.2 Datapath

GEZEL models synchronous single clock designs, however clock signal is not present in the language. Clock is implicit in design description. It is implemented by the use of two variable types in GEZEL: signals and registers. Signal holds its value only within single clock cycle, while register stores its value over multiple clock cycles. Register has two values: next value at the input and current value at the output. Before next clock cycle, current value is update with the next value. This feature of register allows clock-cycle

true description without defining the clock explicitly. Each register and signal has its name and type. Initial value for register is zero while for signal is undefined.

Datapath consists of:

- Set of inputs and output ports, which are the only way the datapath can be accessed from outside.
- Signals and registers - local variables that can be accessed only from inside of the datapath
- Set of signal flow graphs that define operations on the datapath elements.

Hierarchy is possible in GEZEL. Datapaths can be included inside other datapaths with the *use* keyword, this construct is presented on the figure 2.5. Signals of the parent datapath are connected to the input and output ports of the datapath that is being used.

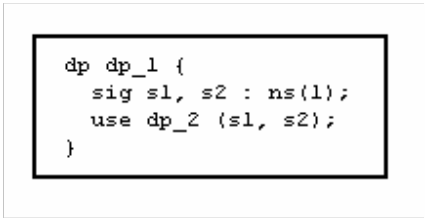

Figure 2.5: Use syntax.

GEZEL provides possibility for cloning, in case when multiple copies of a single datapath are needed. Each datapath obtained this way is an independent copy. Listing for cloning operation is presented in the figure 2.6.

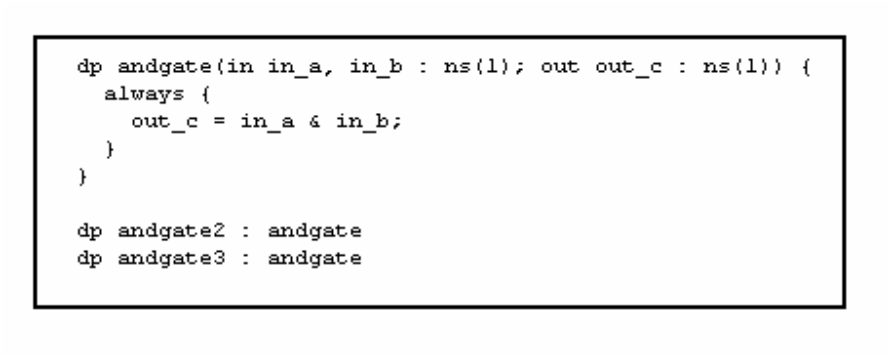

Figure 2.6: Cloning of datapaths.

2.3 Signal flow graph

Signal flow graph is a group of operations on datapath variables that are executed together in a single clock cycle. Number of expressions is arbitrary. The order of expression execution is not depended on the order they are defined in the program, it depends on the precedence of signals and registers. For registers there is always available

value to read, however it is not the case with signals. Signals store value only within single clock cycle, so its value has to be written before it can be read.

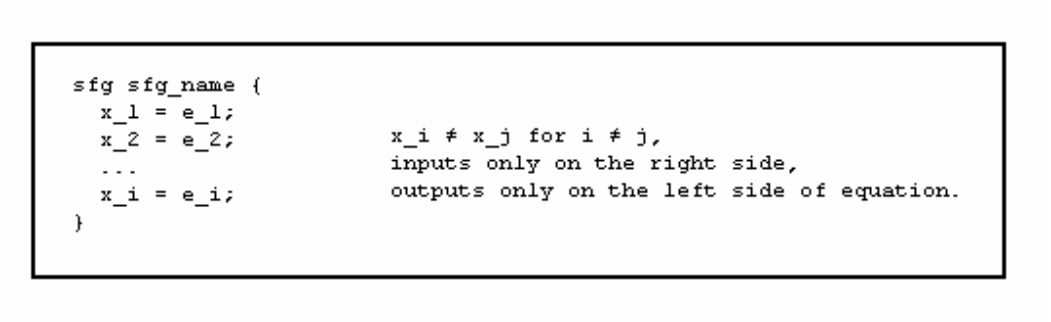

Figure 2.7: Signal flow graph syntax – definition of a single action.

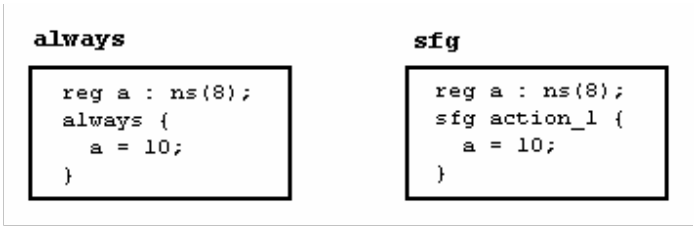

Figure 2.8: Signal flow graph definitions.

There are two types of signal flow graphs in GEZEL, figure 2.8:

a) Unnamed, created with keyword *always*, which is executed every clock cycle. Only one signal flow graph of this kind is allowed within a single datapath.

b) Of specified name, created with keyword *sfg*, scheduled by the controller.

2.4 Rules for datapath

As specified in [1], there are four rules to which datapath program must conform:

a) During any clock cycle, all datapath outputs are defined.

b) During any clock cycle, no combinatorial loop between signals can exist. This happens when there is a circular dependence on signal values, i.e. signal a is used to define signal b, and signal b is used to define signal a. This implies that all signal values will eventually only be dependent, during any clock cycle, on datapath inputs, datapath registers and constant values.

c) If an expression uses the value of a signal during a particular clock cycle, then that signal must also appear at the left-hand side of an assignment expression in the same clock cycle.

d) Neither registers, nor signals or datapath outputs can be assigned more than once during a clock cycle. A special case of this is that a datapath input cannot be assigned inside of a datapath, because a datapath input must be driven by the output of another datapath.

2.5 Controller

Controller schedules execution of the named signal flow graphs. Datapath name to which it is attached is always specified in its declaration. There are three possibilities in GEZEL language provided for definition of controllers:

a) Hardwired, which during each cycle executes all signal flow graphs specified in its declaration. Figure 2.10.

b) Sequencer, each cycle executes only one signal flow graph from the list, picking the one that is supposed to be invoked in a cyclic fashion. Figure 2.10.

c) Finite state machine controller, combines execution of actions together with decision making. Declaration of this controller consists of finite set of possible states with one of those states declared as initial one. During each clock cycle there is exactly one current state of the controller.

Unnamed signal flow graph *always* is not scheduled by controller and it is executed during each cycle.

Finite state machine controller consists of set of possible states, with specified initial one and declaration of transitions. Transition which will be taken by the controller in the next cycle is fully deterministic and depends on the current state and condition imposed on the values of datapath registers. Determinism of the transitions is forced by the use of *if – else* structure in declaration which covers all possible transitions in a distinct cases. Figure 2.9 shows transition definition scheme. During transition a specific action is performed - all signals flow graphs listed in transition declaration are executed. Finally, controller changes its current state to the one declared as a next state for this transition.

```
@state_1 if (condition_1) then (action_1) -> state_1;
    else if (condition_2) then (action_2) -> state_2;
    ...
    else if (condition_n) then (action_n) -> state_n;
    else (condition_m) -> state_m;
@s2 (outidle) -> s2;
```

Figure 2.9: Example of controller transition declaration in GEZEL.

GEZEL language offers several shorthand notations for controller definition, like hardwired or sequencer. Each of those constructs represents a simple controller. Declaration of those controllers may be seen on the figure 2.10. Hardwired controller executes actions 1 and 2 every cycle, while sequencer during one clock cycle selects single action from the list at the time - it executes them in a cyclic fashion.

```
hardwired controller_1(datapath_1) { action_1; action_2; }

sequencer controller_2(datapath_2) { action_1; action_2; action_3; }
```

Figure 2.10: Hardwired and sequencer controllers.

3 A graphical tool for modeling and analysis

In this chapter, a tool will be presented, which can be used for development and verification of hardware architectures implemented with GEZEL-like languages. The tool provides the user with graphical environment, where he can in an easy way create, view and modify his models. Some basic verification procedures are integrated in the tool, so the user is instantly notified if particular kind of errors is present in the model. Moreover, it is possible to generate NuSMV program that behaves exactly like currently developed model, which further allows the user to conduct full functional check, by confronting generated program against logical specifications formulated in Computation Tree Logic or Linear Temporal Logic.

Structure, that represents model in the tool, is based on the concept used in GEZEL. Basic building block of the system is module. There is one root element – system module. Modules create hierarchical structure. Each module consists of datapath and controller.

Datapath contains:

- List of basic elements: inputs, outputs, signals, registers.
- Signal flow graphs, which are sets of operations on basic elements.
- Instances of other modules which are used as sub-module by this module, with defined connection between sub-module inputs, outputs and parent module signals.

Controller contains set of states. Each of those states has its own list of possible transitions. In every transition there are defined: condition, list of signal flow graphs which are executed and next state.

3.1 User interface

When application is started, initial screen is presented to the user – figure 3.1. Now, user can either load existing project, import XML file, parse GEZEL program or edit the model by hand.

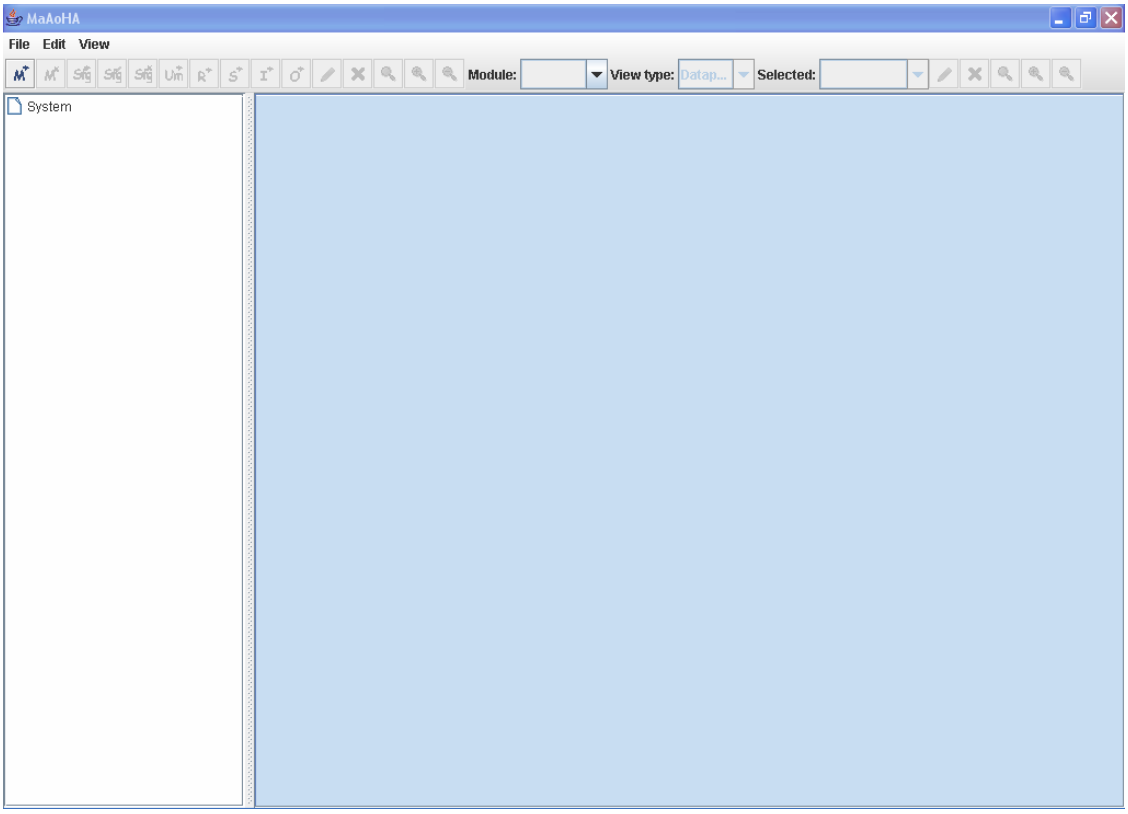

Figure 3.1: Starting screen.

3.1.1 Menu

All read/write options are accessible from the file menu. The menu may be seen on the figure 3.2. There are several possibilities, for model importing and exporting, accessible. Basically, there are three types of files that can be used for this purpose:

- XML model
- XML model with project settings
- GEZEL program

It is possible to create XML files both with and without data used in graphical representation, like position of vertices. In case when it is chosen not to use project settings only pure logical model of the system is taken into consideration. Except XML files, GEZEL sources can be used both as input and output. The tool is able to parse GEZEL code and create corresponding model. The structure used to represent the model by the tool does not include elements for all concepts from GEZEL language, for example, sequencer and hardwired controller are such concepts. However the tool is able to create controller that corresponds to those two and behaves exactly in the same way. Finally, there is an option for creation of NuSMV model corresponding to the current

model, which can be later used in full functional check of the system. This issue will be discussed in more details in the chapter 4.

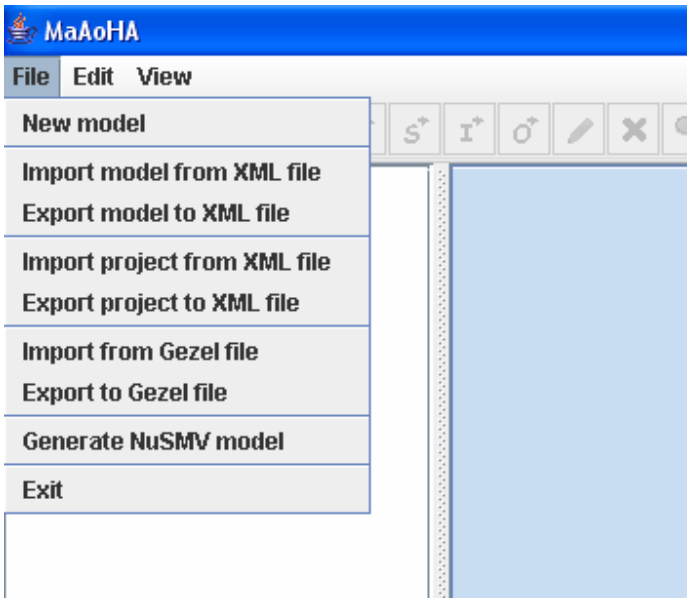

Figure 3.2: Menu file.

3.1.2 Main view

If the user decides to import a model from file, all windows of the tool are being populated with a graphical representation of the model. Example of the model view created by the tool is presented on the figure 3.3.

Tool view consists of three main parts. On the left side model is presented in a form of tree structure. The root of the tree contains all modules that make the system. Each module is separated in its own branch and clearly described. Module is divided into two parts: datapath and controller. In datapath part, all datapath elements are listed and divided into several categories: inputs, outputs, signals, registers, signal flow graphs and sub-modules. Input, output, signal and register branches contain definitions of elements of particular type. Their names and types are specified in there. Signal flow graph branch contains leafs representing all expressions that are part of it. Finally sub-modules branch consists of all modules embedded inside current module, with name, kind of sub-module and parameters displayed as a label. The main task of this tree structure is to give the user a good sense of general view of the model, as well as to make navigation, in other components, more convenient, since selection made in here, results in more detailed view presented in the right side window, where datapath and controller are presented in the form of graphs. An example of the tree structure may be seen on the figure 3.4.

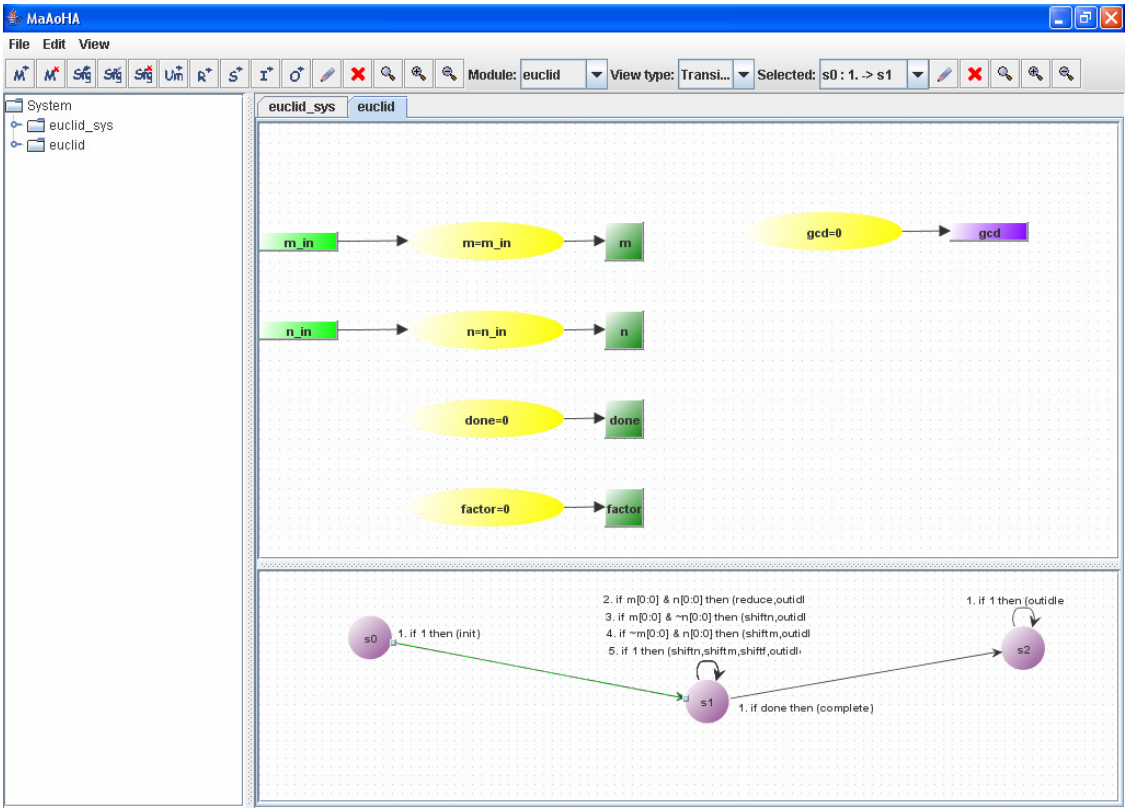

Figure 3.3 Model view.

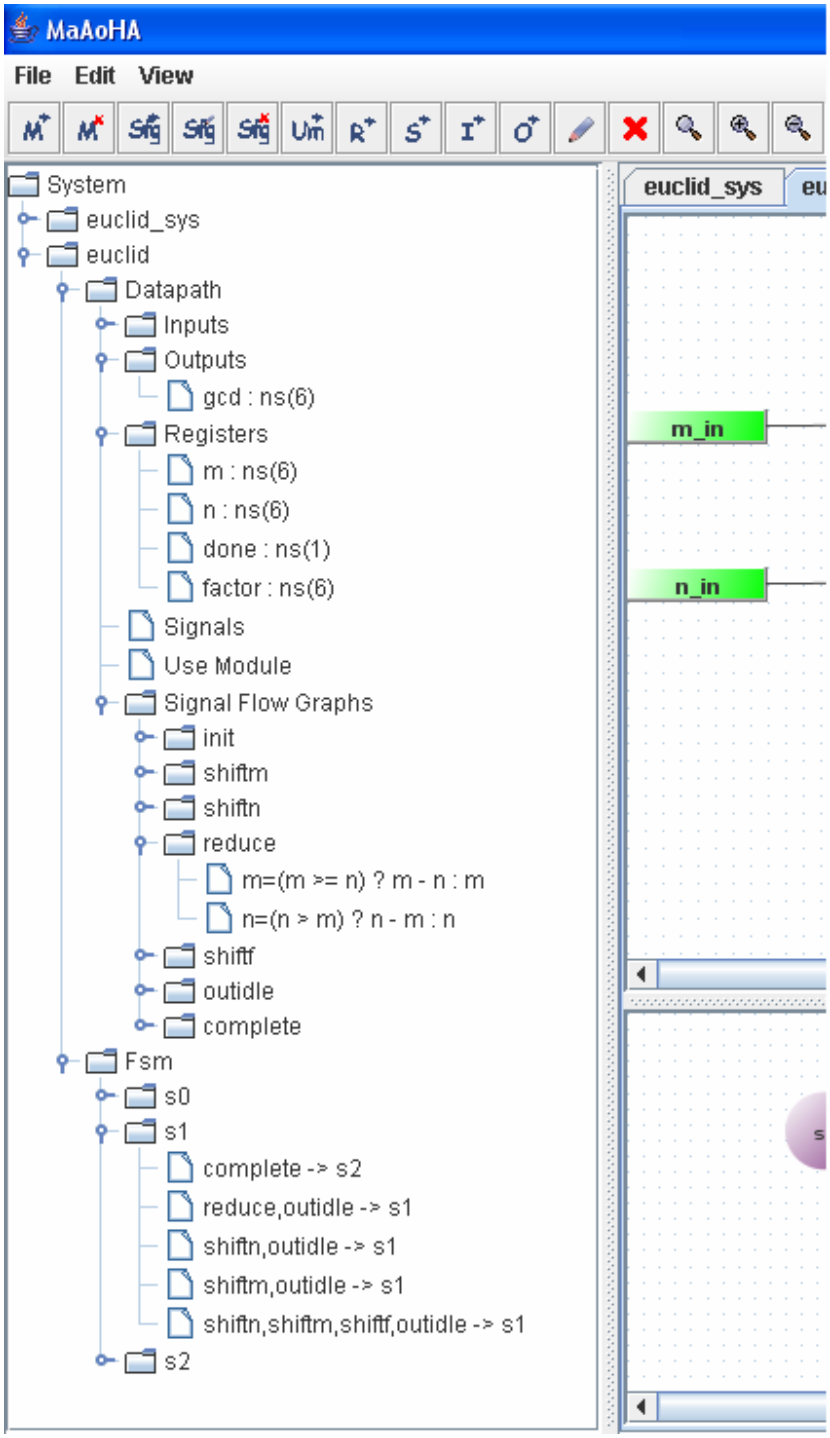

Figure 3.4: Tree structure.

3.1.3 Datapath and controller graphs

Top right part of the view is occupied by the graph representing datapath of the currently selected module. The concept behind graphical representation of datapath components is as follows: inputs, outputs, registers, signals, single expressions from signal flow graphs and instance of sub-modules are represented by vertices. Each of those element kinds is depictured by vertices that differ in size, shape and color. To point out interaction between elements, edges are used. From every expression vertex, there are edges that go to each of the variables (inputs, signals, registers, outputs) used in this expression. If the variable appears on the left side of the expression, an edge is directed towards the variable vertex, otherwise towards expression vertex. Similarly for sub-modules, edges connect associated vertex with the variables present in *use* declaration. If a variable is used as an input, the edge is directed towards datapath vertex, if as an output, towards the variable vertex. Datapath graph may be seen on the figure 3.5.

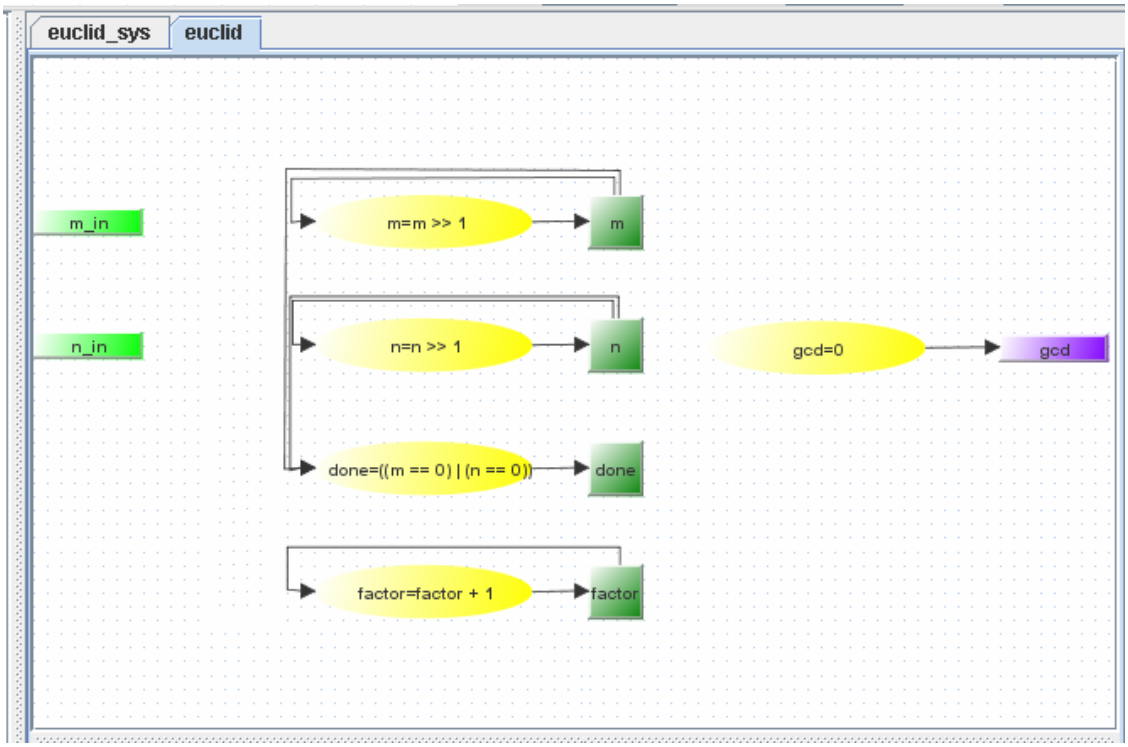

Figure 3.5: Datapath view.

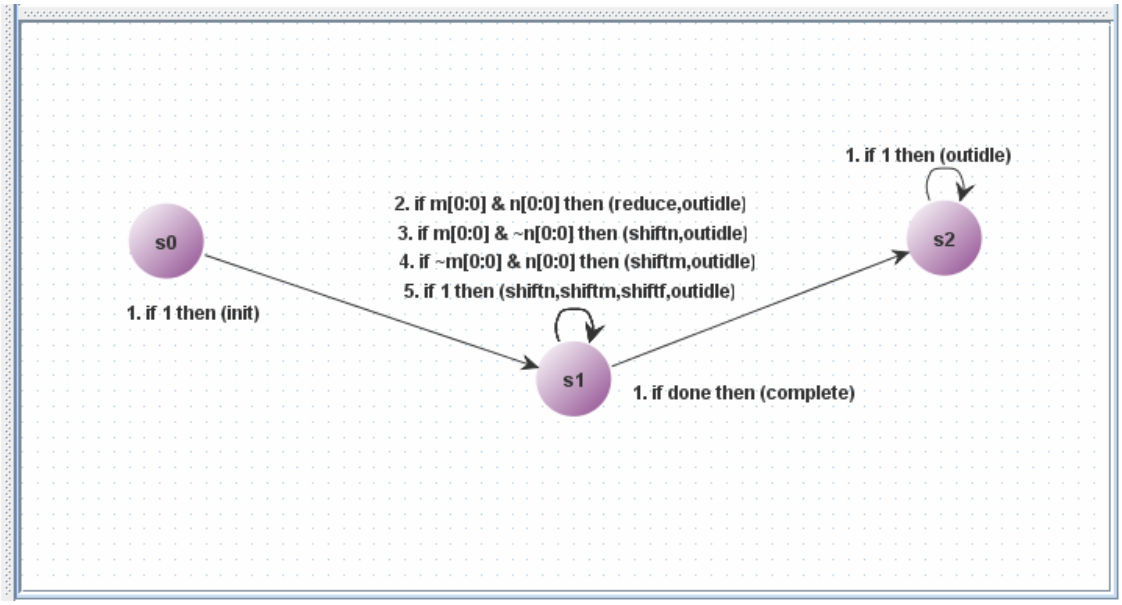

Figure 3.6: Controller view.

User is able to choose one of several available views of datapath. Generally those view fall into three categories: general view, signal flow graph view and transition view. In general view, the only displayed elements are inputs, outputs, registers, signals and sub-modules. Briefly speaking, components that are not specific for any particular signal flow graph. This view does not present any signal flow graph thus no expressions are present. Second kind of view is signal flow graph view, except generic datapath components, it pictures expression present in a single signal flow graph. Finally, the last kind of view – transition view, combines generic components with expressions included in all signal flow graphs that are invoked during the specific transition.

Controller graph, placed in the bottom right corner of the tool view, consists of two kinds of elements, states represented by vertices and transitions represented by edges. Transition edge is labeled with its index, condition and signal flow graphs that are launched during its execution. Controller graph is presented on the figure 3.6.

3.2 Navigation

A model can have many modules. Every module can have dozens of signal flow graphs and transitions, each with its own separate view. It is very important to make navigation between all those different views easy and intuitive for the user. For this purpose, there are several possibilities for view change implemented. At the toolbar user can find three dropdown lists. First one contains list of all modules of the system, second - type of view (there are three types: general datapath view, signal flow graph view and transition view). Contents of the last one depends on the selected type of view, and can be either deactivated if general datapath view is chosen, filled with all signal flow graphs or all

transitions of the selected module. Apart of letting the user know, what kind and which view in particular he is facing, those three drop down list allow change, in a fast and easy way, to any view of the model that user may want to see at the moment. The user can change view also in two other ways. First way to do that is to perform selection on the tree structure. In case when selected element is a datapath, signal flow graph or transition node appropriate view will be displayed and three dropdown view-lists will be updated to show properly, which view is selected. The last way of choosing view is to point a transition on the controller graph, which will automatically switch current view to view specific for this transition. All elements that provide change of view functionality are presented on the figure 3.7.

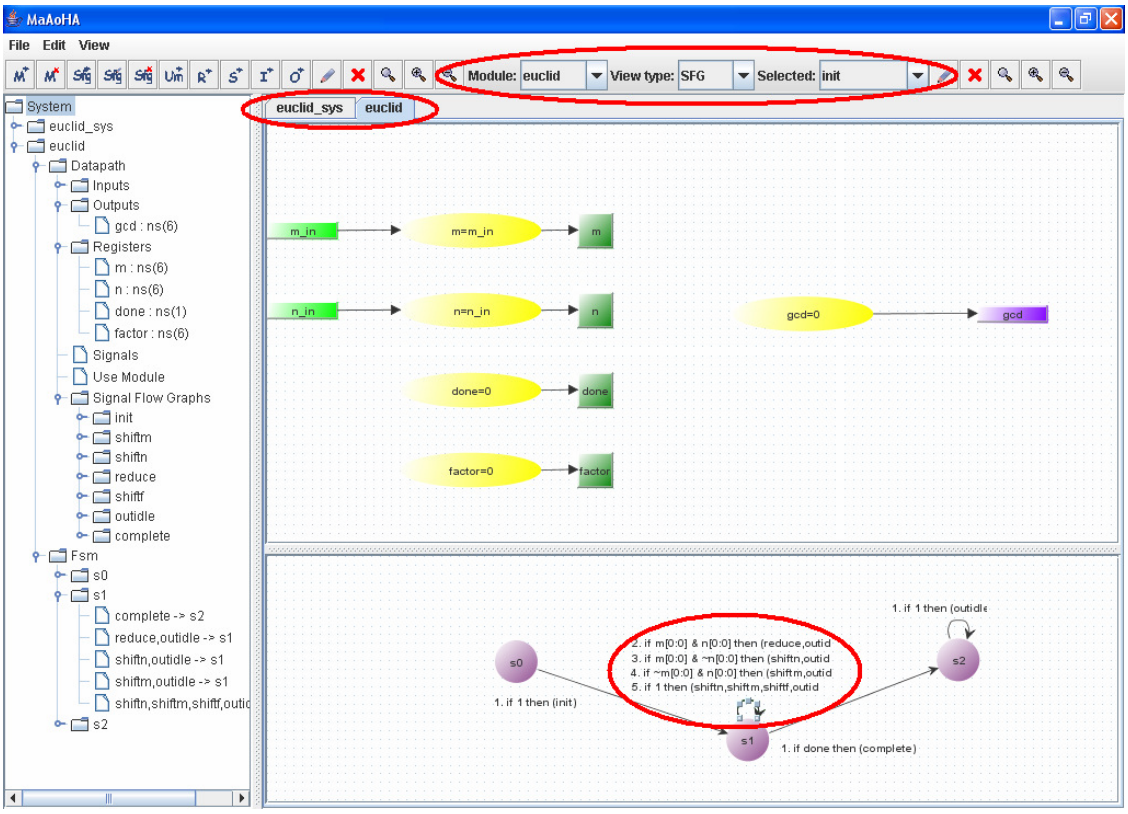

Figure 3.7: Elements that provide change of view functionality.

3.3 Model editing

The tool provides user with possibility to create a model from scratch by hand or edit imported one. In order to do that, there are several functions available. Those functions may be accessed from the menu edit, presented on the figure 3.8. It is also possible to reach modification procedures by pressing appropriate button on the toolbar or by use of popup menu, which appears after right click of the mouse anywhere in the graph window,

both for datapath and controller. All elements of the tool that provide model edition functionality may be viewed on the figure 3.9.

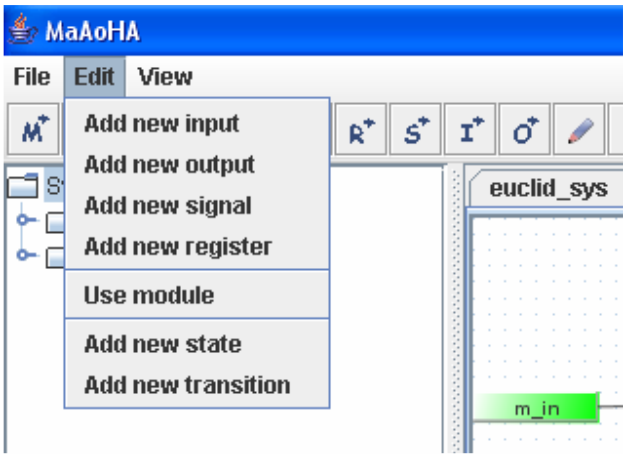

Figure 3.8: Edit menu.

User is able to:

- Add and remove module
- Add, edit and remove signal flow graph from any existing module
- Add input, output, signal, register and sub-module to datapath
- Add expression to signal flow graph.
- Add state and transition to controller

Moreover, by selecting one or more elements visible on datapath or controller graph, user can delete or edit those. Each of editing functions, except removal, which instantly deletes selected elements, launches modal dialog where all necessary details have to be specified.

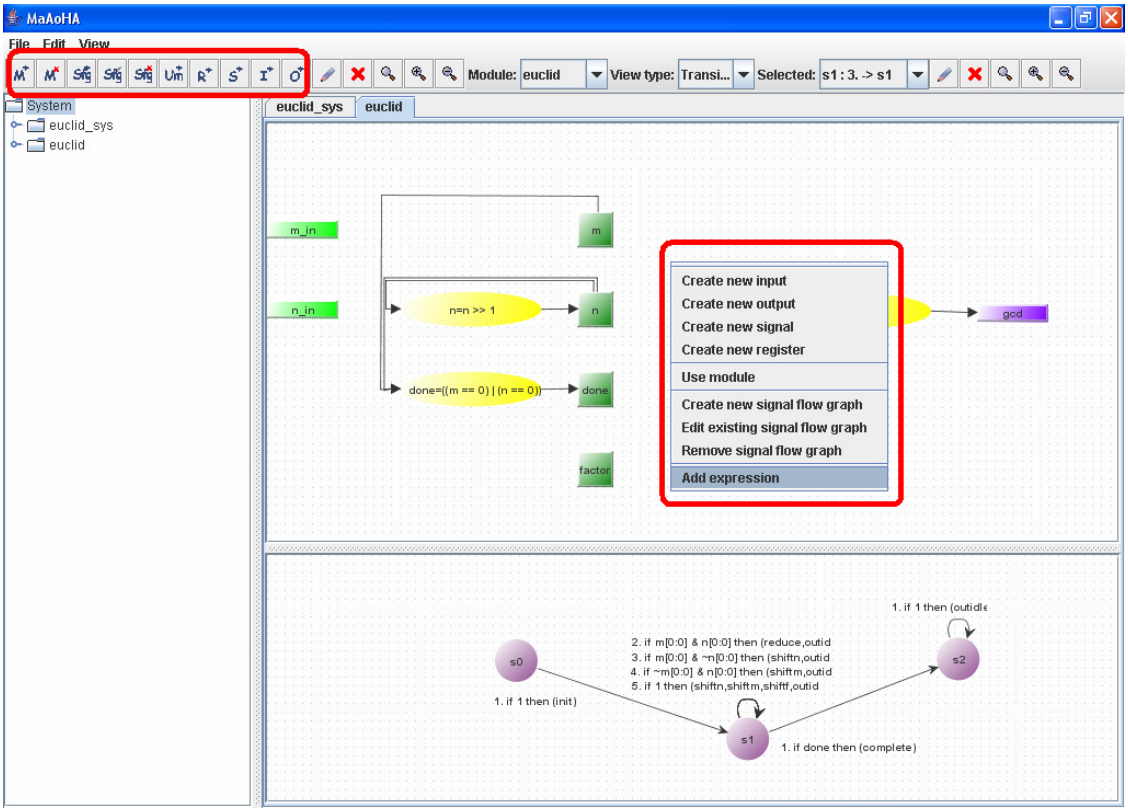

Figure 3.9: Elements that provide functionality for model edition.

3.4 Well-formedness check

There are several rules that each model for GEZEL-like language must obey. Those rules are specified in the chapter 2 of this report. In order to assist user in the development process, after each edit operation, that imposes any change in the structure of the model, there is basic check performed. If this verification procedure finds out that some part of the model happens to be incorrect, then appropriate element is point out to the user, by marking with red color the vertex or the edge associated with it. Result of verification of an incorrect syntax may be seen on the figure 3.10.

However, the check is performed on the whole model after each update, there are several scopes what kinds errors are presented to the user. This issue is highly connected with the selected view type. For example, suppose there is a following error: in two signal flow graphs the same variable is assigned and there is a transition which invokes both signal flow graphs, then expression cells, which duplicate calculation of the same variable, will be marked as faulty only in the transition view, not in the signal flow graph views, because at the level of the signal flow graph there is no error. Errors that are present in a lower scope are always visible in all higher scopes. For example if there is an error that two sub-modules assign the same signal variable, situation presented on the figure 3.11, it

will be pointed out to the user in every signal flow graph and transition view of the module for which this error occurred.

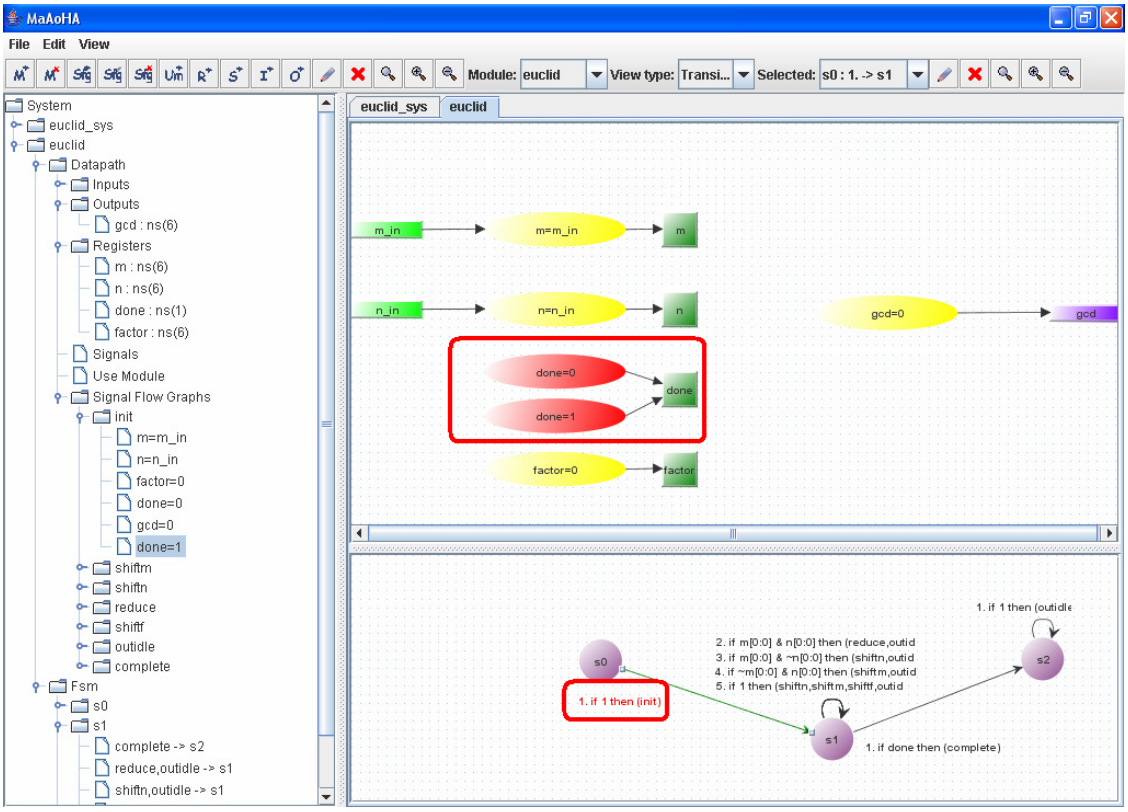

Figure 3.10: Syntax check – multiple assignments of a single variable.

Complete list of errors with its scope specified looks as follows:

Datapath scope:

a) Incorrect parameters for sub-modules.
b) Multiple assignments of the same variable by sub-module.

Signal flow graph scope:

a) Incorrect elements in expression definition – occurs when a variable used in the expression was removed from datapath.
b) One variable is calculated by multiple expressions in a single signal flow graph.

Transition scope:

a) Multiple assignments of the same variable in a signal flow graphs launched by the same transition.
b) Signal is used as a right side element of the expression, which is not assigned anywhere in signal flow graphs launched by the transition – dangling signal.

Absence of combinatorial loops in expression calculations is not verified, during this well-formedness test, by the tool itself. Possible solution to provide this kind of verification would be to create a graph that models dependencies between expressions and check, using graph theory, if loops are not present.

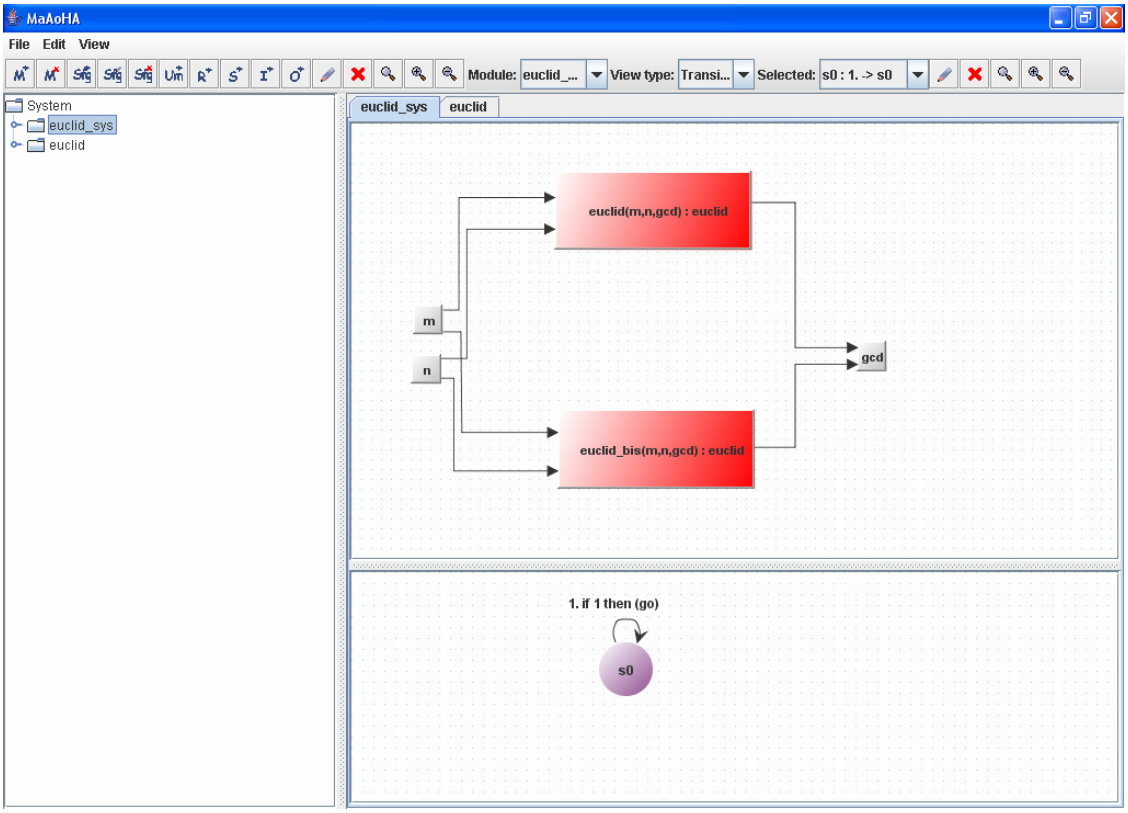

Figure 3.11: Syntax check – Use of a single signal as a parameter for multiple sub-modules.

3.5 Example - Euclidean algorithm

This section presents an example of hardware description – Euclid algorithm, visualized by the tool.

3.5.1 Program structure

The system consists of three modules:

- *euclid_sys* module, which encapsulates and composes *euclid* and *test_euclid* modules.
- *euclid* module that calculates greatest common divisor of two numbers according to the Euclidean algorithm.
- *test_euclid* module, which provides input values.

The complete model structure may be observed in the figures 3.12 and 3.13.

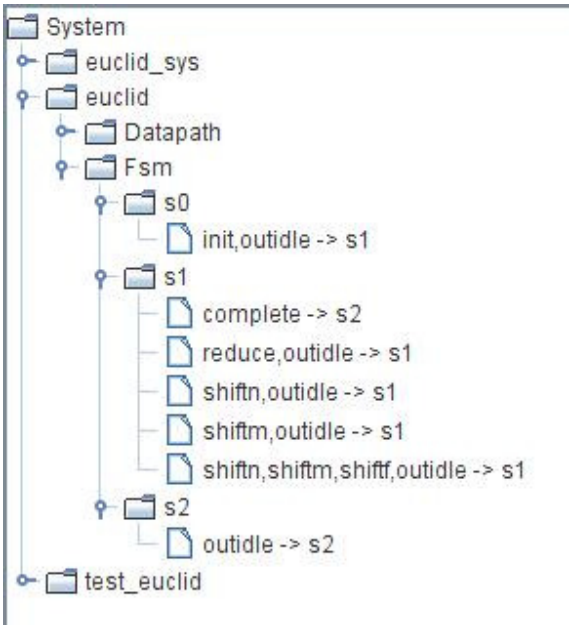

Figure 3.12: The tree structure for Euclid algorithm – part 1.

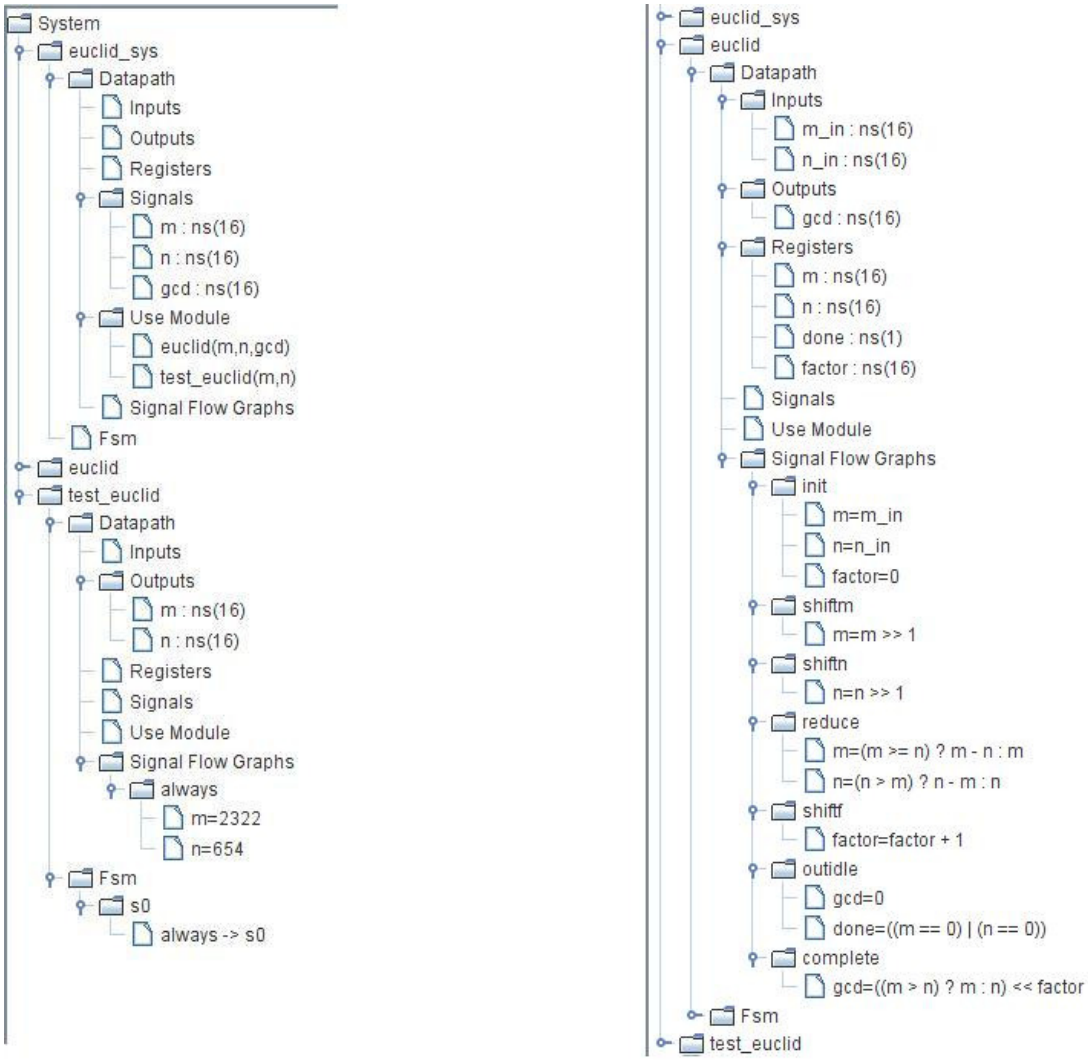

Figure 3.13: The tree structure for Euclid algorithm – part 2.

3.5.2 System module

Main module of the system is very simple and contains three signals and instances of *euclid* and *test_euclid* modules. Signals *m* and *n* connect *test_euclid* output ports with *euclid* inputs, while signal *gcd* is used for storage of the calculation result. The *euclid_sys* module is presented on the figure 3.14.

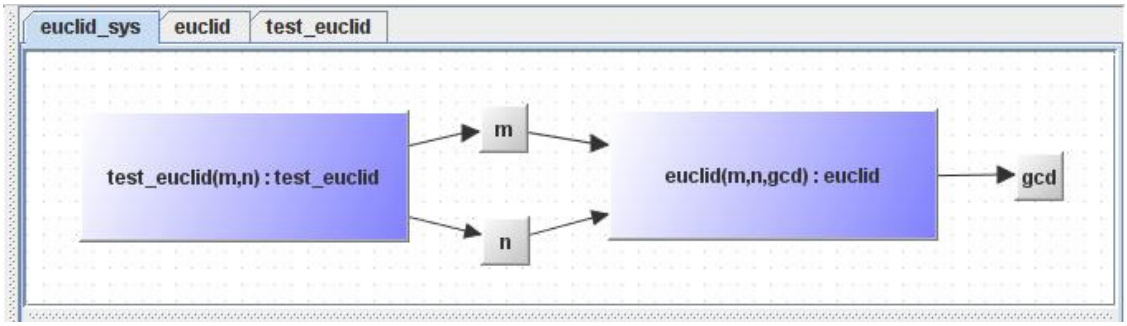

Figure 3.14: The euclid_sys module.

3.5.3 The euclid module

This module is an actual implementation of the Euclidean algorithm. The controller consists of three states: initial state s0, state s1 in which GCD calculations take place and final state s2. The controller is presented on the figure 3.15. The datapath except input and output ports contains four registers. Registers *m* and *n* store values that correspond to input variables, allowing its reduction according to the algorithm flow. Register *done* is a flag that signalizes whether the algorithm has completed. Register *factor* indicates value of the GCD, which is a product of one of the variable registers after reduction and factor of two determined by this register. In the first cycle input values are read into datapath's registers *m* and *n*, and the controller traverses to the state s1. It remains in this state until one of the variables stored in the registers *m* and *n* is reduced to zero. There are several different reduction types, depending on the values of the registers *m* and *n*, each one invoked by the separate transition. After reduction phase, GCD is calculated using values of the *factor* register and the variable register that is greater then zero. Finally, the result is written in the output port and program enters idle state.

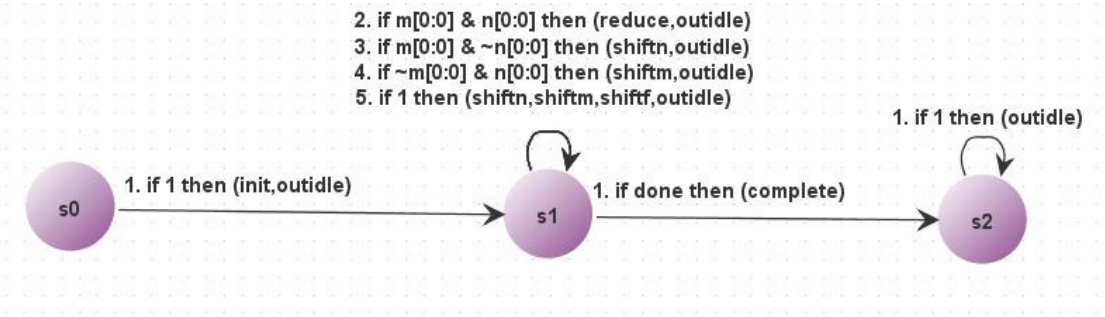

Figure 3.15: Controller of the euclid module.

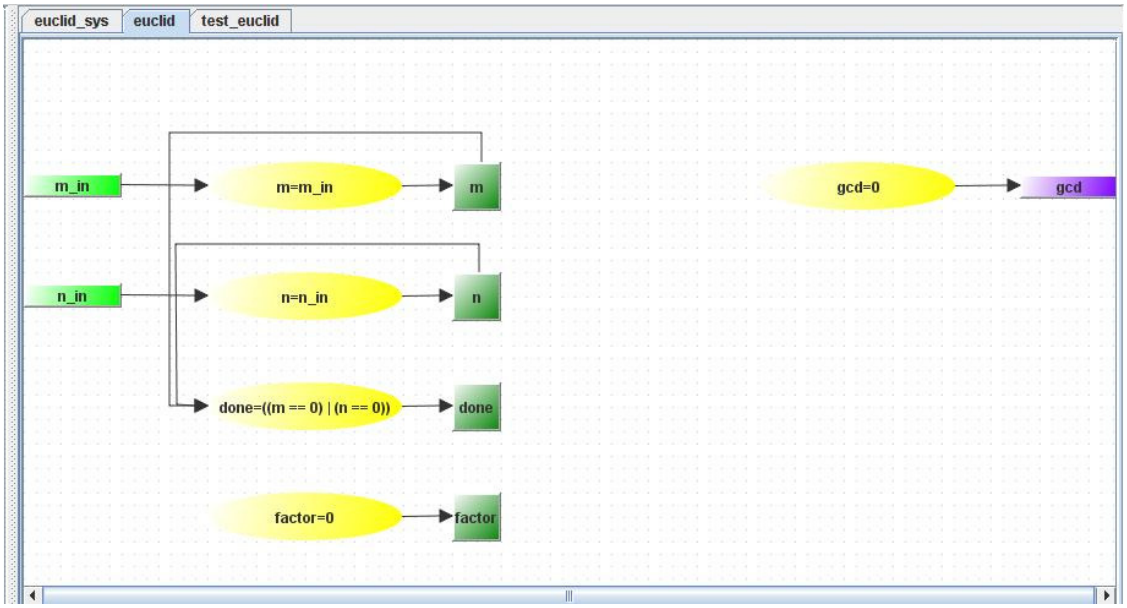

Figure 3.16: Transition view for transition from state s0 (init, outidle).

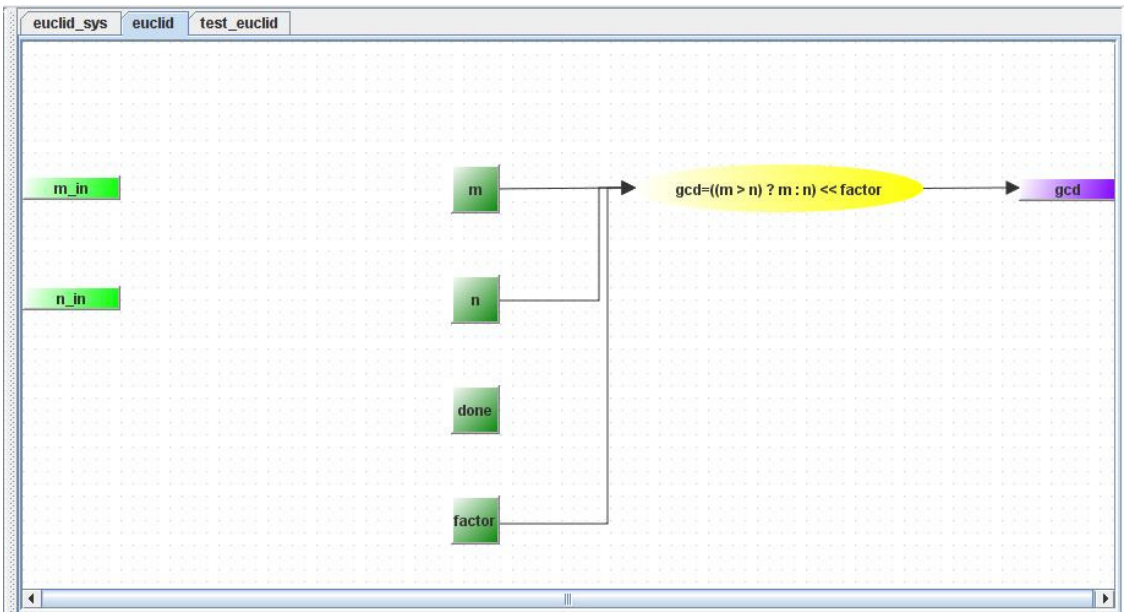

Figure 3.17: Transition view for transition number 1 from state s1 (complete).

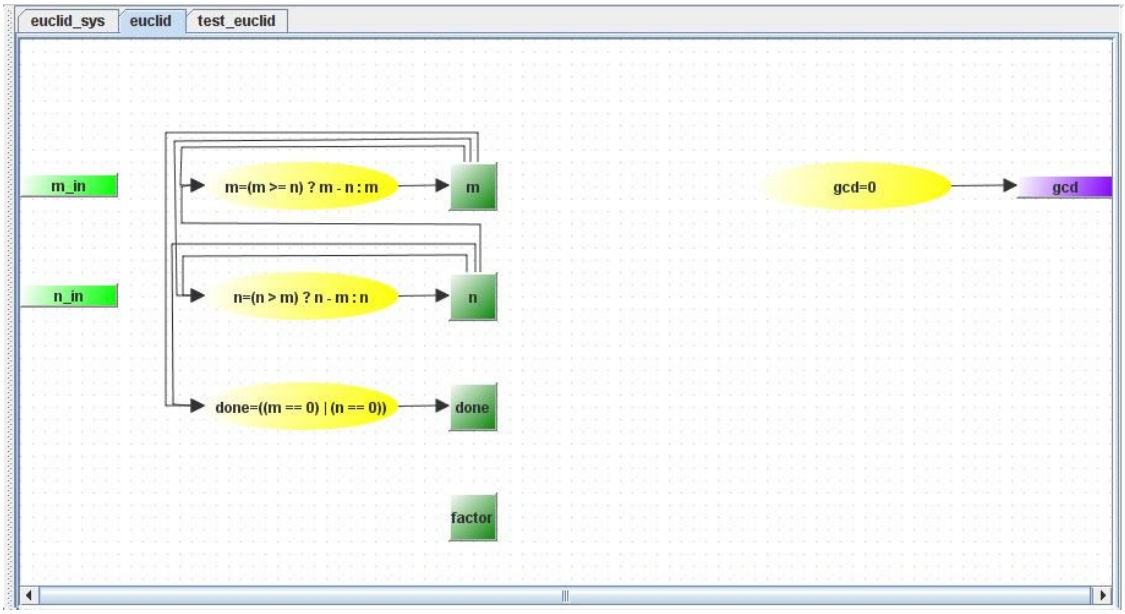

Figure 3.18: Transition view for transition number 2 from state s1 (reduce, outidle).

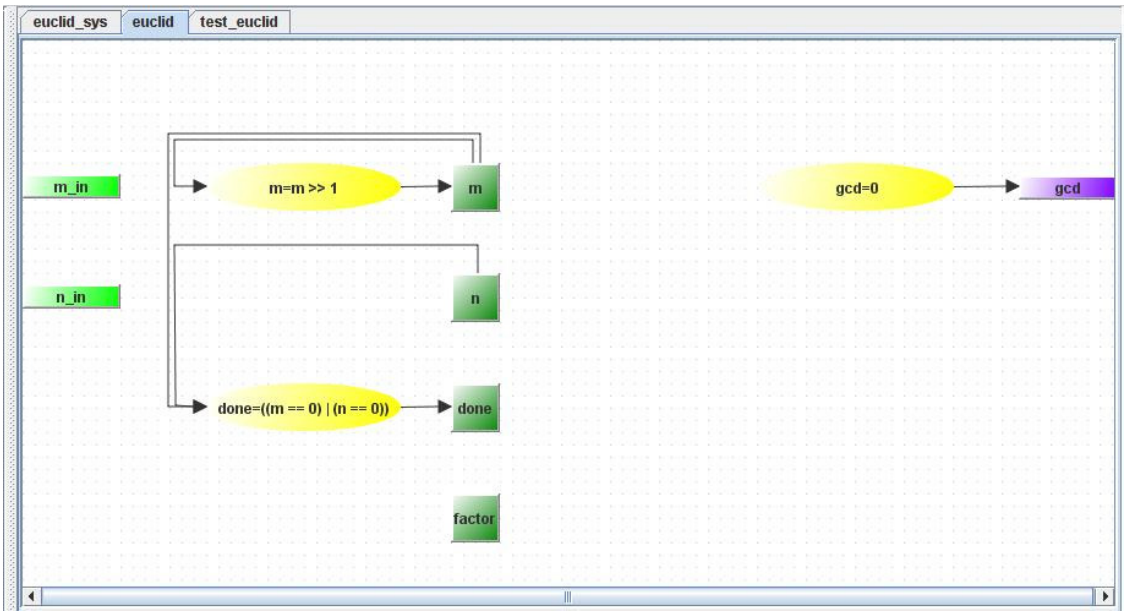

Figure 3.19: Transition view for transition number 4 from state s1 (shiftm, outidle).

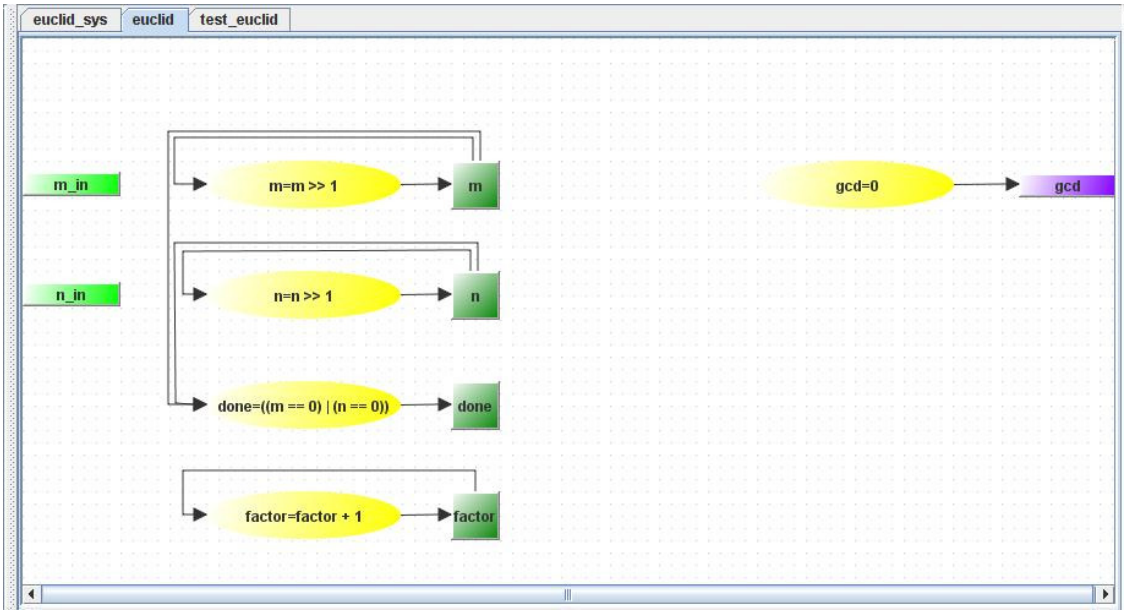

Figure 3.20: Transition view for transition number 5 from state s1 (shiftn, shiftm, shiftf, outidle).

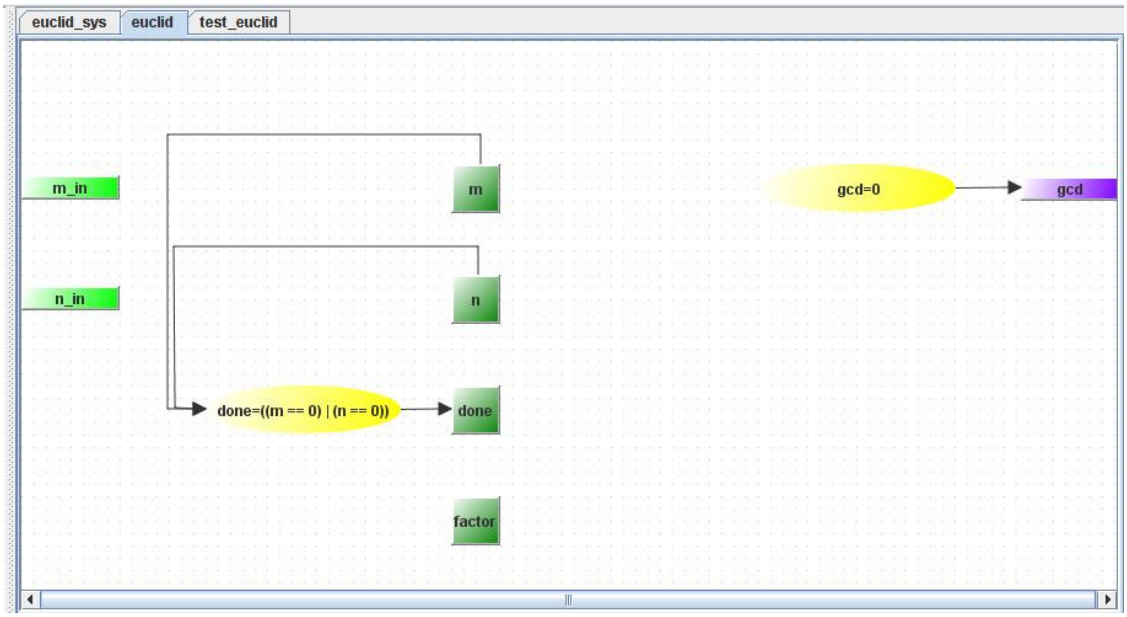

Figure 3.21: Transition view for transition from state s2 (outidle).

3.5.4 The test_euclid module

The purpose of this module is to provide system with an input values. In the controller there is only one state and one transition that is executed during each cycle. The transition

invokes signal flow graph *always*, where output ports have assigned constant values. The controller and the datapath for this module are presented on the figures 3.22 and 3.23.

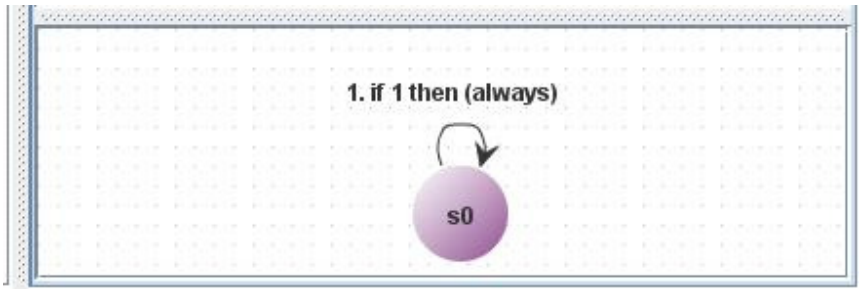

Figure 3.22: The test_euclid controller.

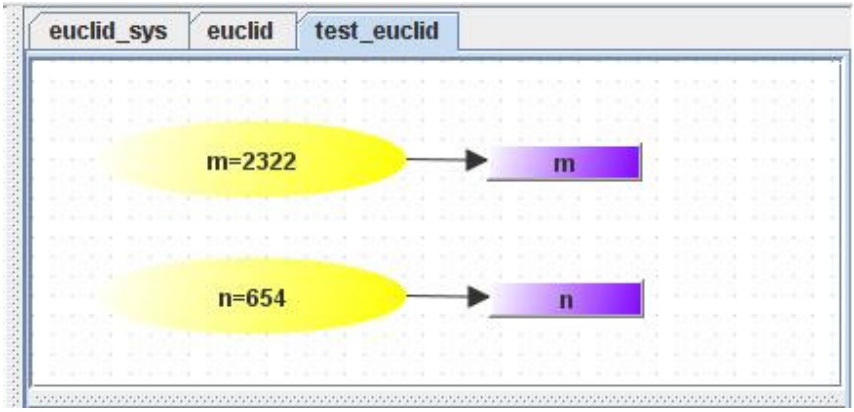

Figure 3.23: Signal flow graph always of the euclid_test datapath.

3.6 Example - Simplified Data Encryption Standard

In this section, a model of the Simplified DES algorithm, developed using the tool, is presented. The system works as follows. The test module generates an input plain-text and a key for the algorithm, which are read by the encryption module and ciphered according to the SDES algorithm. The cipher-text and the key are read by the decryption module and deciphered. Finally, obtained result is the plain-text that was provided by the test module in the first place.

3.6.1 Program structure

The complete structure of the system in the form of tree may be seen on the figures 3.24, 3.25, 3.26 and 3.27. The model consists of four modules:

- system module
- encrypt
- decrypt
- test_encrypt

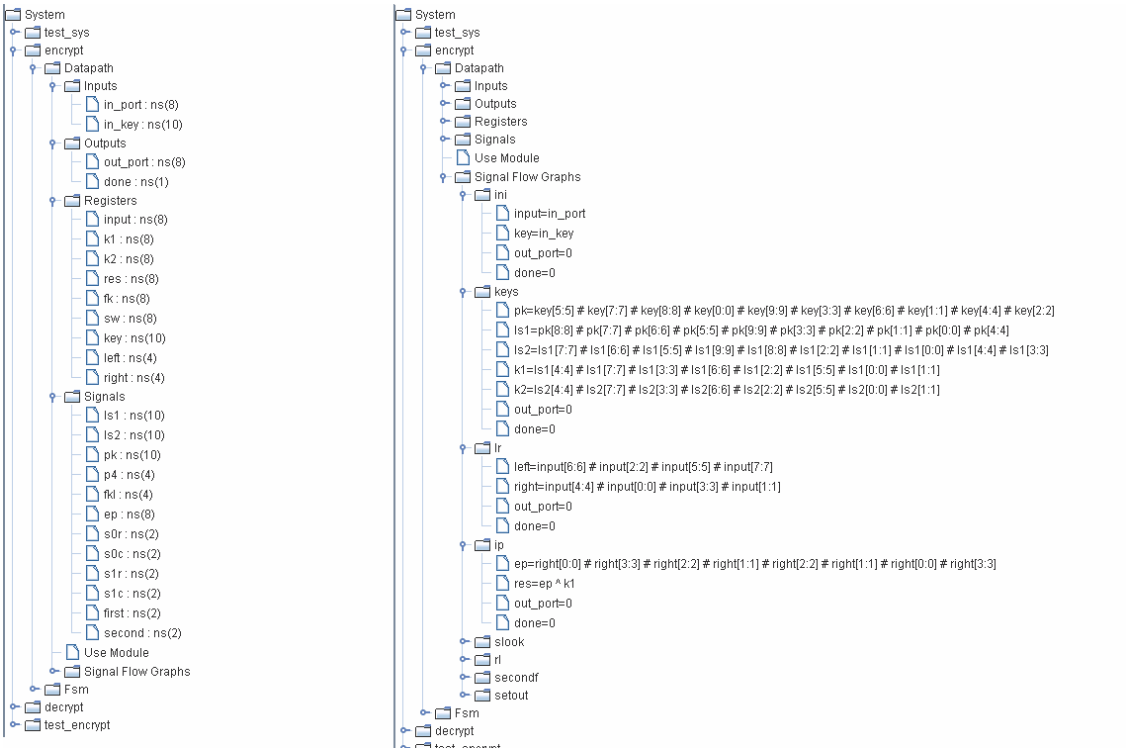

Figure 3.24: Tree structure of the SDES algorithm – part 1.

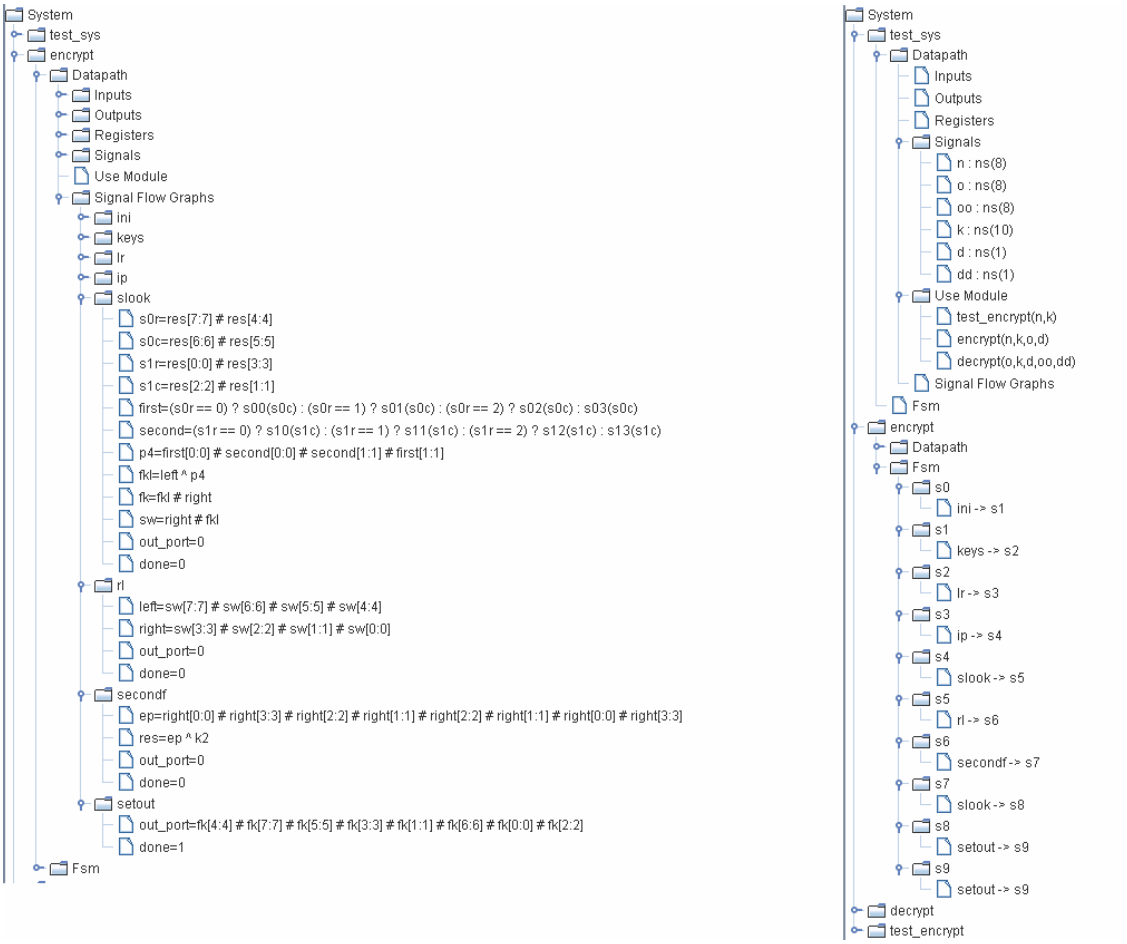

Figure 3.25: Tree structure of the SDES algorithm – part 2.

Figure 3.26: Tree structure of the SDES algorithm – part 3.

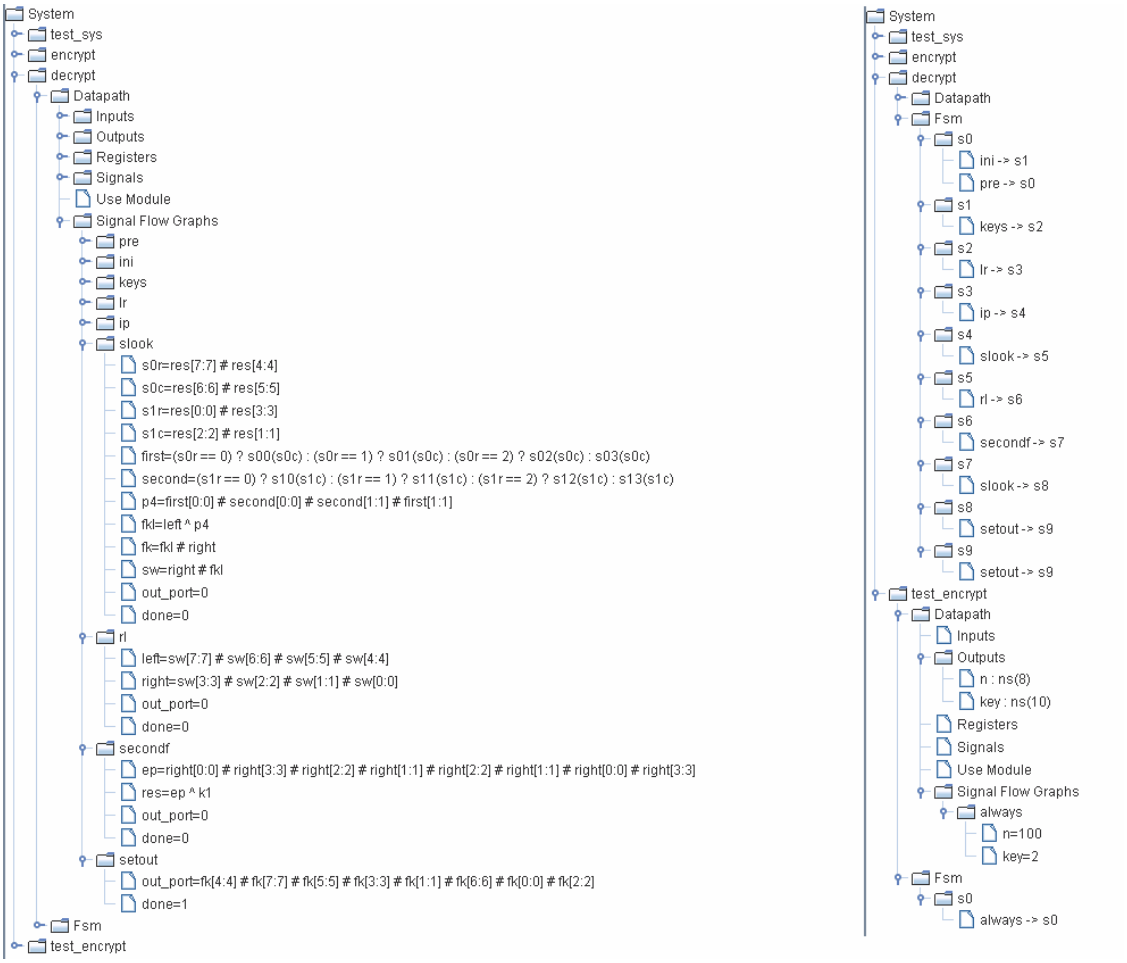

Figure 3.27: Tree structure of the SDES algorithm – part 4.

3.6.2 System module

In the main module, one instance of each of the remaining modules is invoked. Those module instances are composed according to the program flow presented before. The *test_enrypt* module output ports that provide the plain-text and the key are connected to the signals *n* and *k* respectively. Input ports of the encrypt module read value from those two signals. The output ports of *encrypt* hold a ciphered-text and a done flag, which indicates whether encryption process has completed, and are connected to the signals *o* and *d* respectively. Finally, the decrypt module reads values from the signals *o*, *d* and *k* and on its output ports passes the values to signals *oo* and *dd*, which store the plain-text and done flag respectively. The complete structure, as it is represented in the tool, may be observed on the figure 3.28.

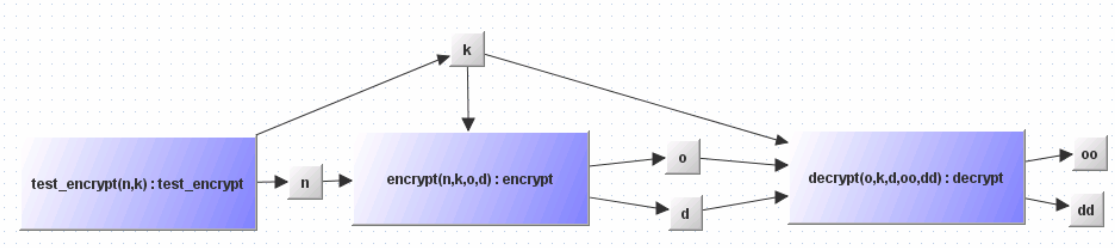

Figure 3.28: Main module of the SDES algorithm.

3.6.3 Encryption

The encrypt module as an input takes a plain-text and a key. During several phases that perform various operations on the inputted values, according to the SDES algorithm, ciphered text is generated and the result written to the output ports.

The controller for this module presented on the figure 3.29 has a simple, linear structure. There are no conditions on transitions or loops. During the first transition the encryption process is initiated and values on input ports are read. Next several transitions correspond to different phases of algorithm, where text modification operations take place. Finally, obtained cipher-text is passed to the output port and the module enters idle state. The datapath contains number of signals and registers that store intermediate values used in the encryption process. Examples of signal flow graph representations for this module may be seen on the figures 3.30 and 3.31.

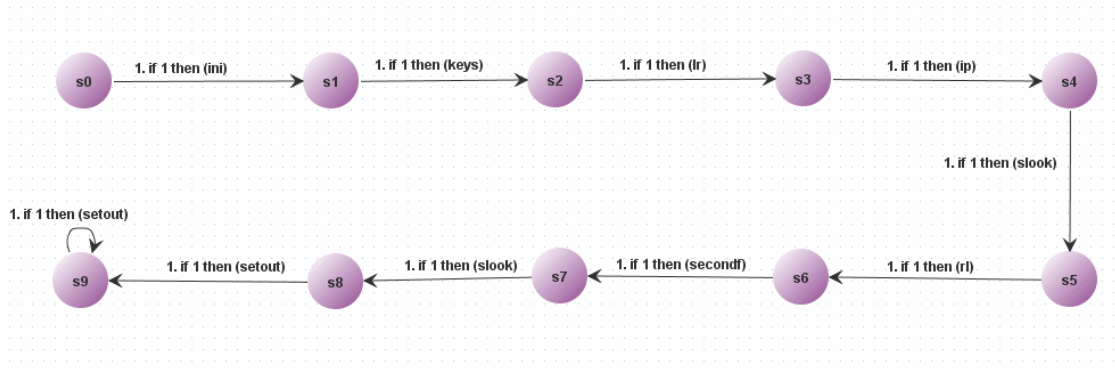

Figure 3.29: Controller of the encrypt module.

Figure 3.30: Examples of the signal flow graph representation for the encrypt module – part 1.

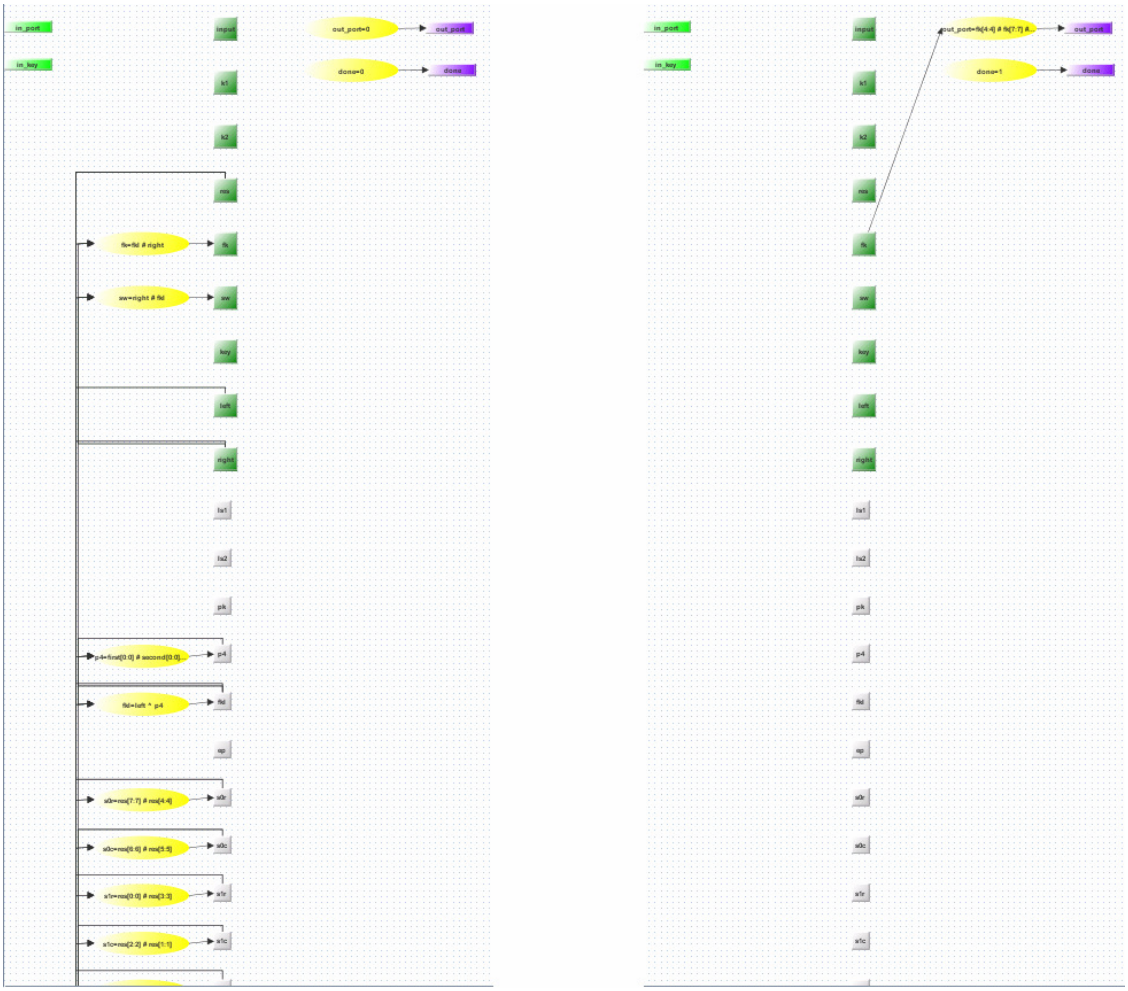

Figure 3.31: Examples of the signal flow graph representation for encrypt module – part 2.

3.6.4 Decryption

The module responsible for decryption does the opposite to the *encrypt* module. On the input ports it reads a ciphered text and a key, and after several phases of algorithm-specific operations for decryption, a plain text is passed to the output ports.

Like it was in the *encrypt* module case, the controller is linear with no conditions or loops. The only exception is initialization of the process, which can start after a cipher-text is generated and the done flag is set by the *encrypt* module. After an encrypted text and a key are read, decryption operations take place, divided into several phases. Finally, before entering the last state, obtained result – a deciphered text is passed to the output port. The controller is presented on the figure 3.32 and sample datapath screenshots for this module on the figure 3.33.

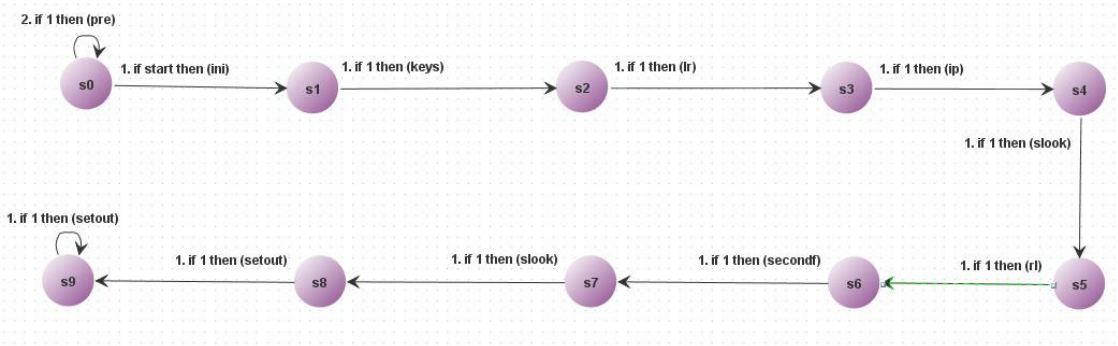

Figure 3.32: Controller of the decrypt module.

Figure 3.33: Examples of the signal flow graph representation for decrypt module.

3.6.5 The test_encrypt module

The module that generates test variables for this model is very simple. Every cycle its output ports, for a plain text and a key, have assigned the same values. The controller and signal flow graph always are presented on the figures 3.34 and 3.35.

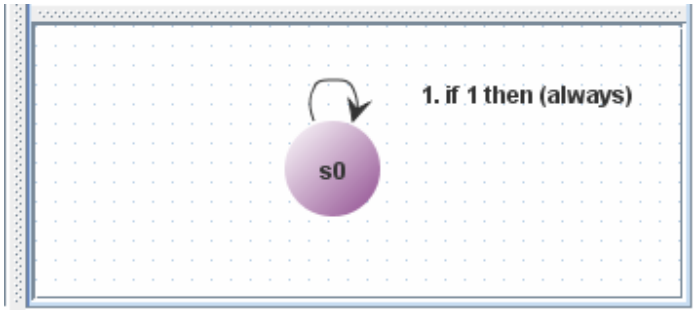

Figure 3.34: Controller of the test_encrypt module.

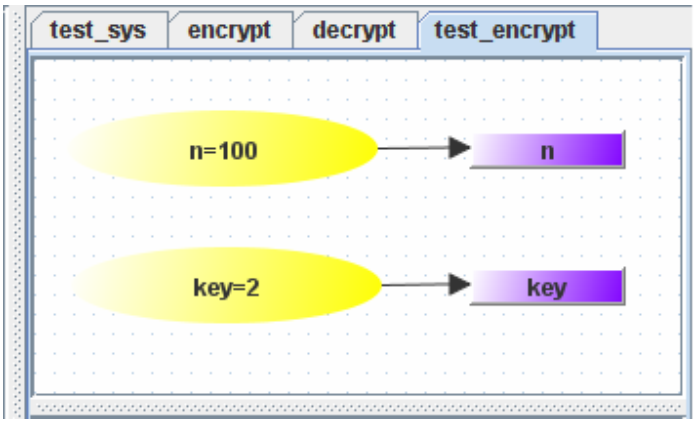

Figure 3.35: The always signal flow graph of the test_encrypt module.

4 Verification using NuSMV

Basic check procedure, implemented in the tool, can verify absence of some errors, but still it does not prove that the model works in the way user desires it to work. However, the tool also provides a possibility to check that. By the use of an external program – symbolic model checker NuSMV, one can perform full functional check of the model. The tool is capable of creation NuSMV program, which models behavior of the currently developed hardware description. NuSMV model obtained this way may be verified using logical formulas specified in Computation Tree Logic or Linear Temporal Logic.

4.1 Introduction to NuSMV

NuSMV is a symbolic model checker based on Binary Decision Diagrams (BDD). The tool was developed by ITC-IRST (Istituto Trentino di Cultura, Istituto per la Ricerca Scientifica e Tecnologica in Trento), Carnegie Mellon University, the University of Genoa and the University of Trento as an extension and reimplementation of SMV, the original BDD based model checker. NuSMV is designed as an open architecture for model checking, aimed at industrially sized designs, a back-end for other verification tools and as a research tool for formal verification techniques. It allows representation of synchronous and asynchronous finite state machine systems and verification of logical formulas specified in one of the following logics: Computation Tree Logic (CTL), Linear Temporal Logic (LTL). In order to do that it uses BDD and prepositional satisfiability (SAT). Heuristics are available for the partial control of the state explosion.

4.1.1 Overview

Input language for NuSMV allows implementation of Finite State Machine (FSM) both synchronous and asynchronous. System specifications may range from detailed to abstract, from Mealy Machines to set of non-deterministic processes. The language provides different description styles. More restricted style reduces danger of inconsistency and appearance of logical contradiction in the system, while less restricted one, allows greater doze of flexibility and makes use of any propositional expression from propositional calculus, in definition of transition relation, possible. All features provided by the language are not covered in this report. The focus is put rather on the parts of the language that are relevant for modeling, in terms of NuSMV, of the system, created using the tool presented in the chapter 3. Therefore, this document provides

introduction to the synchronous systems created by the use of the direct, less restricted description style. In order to obtain more information on the NuSMV language, the reader should refer to [5] and [6].

4.1.2 Program structure

NuSMV program is a hierarchical structure of basic building blocks called modules with one of them defined as the root element. Each module may contain collection of declarations, constraints, specifications and optional list of passed parameters. Declaration part specifies a set of variables and a set of instances of other modules that create the module, while constraint part allows definition of the system behavior through time, by introduction of transition system. A module may be reused, by instantiation inside another module, as many times as necessary.

On the figure 4.1 sample program structure is presented, that corresponds to the GEZEL program from the figure 1.2. Module main has three 6-bit variables of word type declared: *sig_m, sig_n* and *sig_gcd*. Instance of *mod_euclid* module is created with the declared variables passed as parameters. Finally, initial values and transitions for some variables of those are declared. Definition of module *mod_euclid* contains list of parameters that are passed from the parent module.

The root element in the module structure is the one that is evaluated by the interpreter. List of formal parameters is not allowed for this module and its name has to be defined as *main*.

```
MODULE main

VAR

sig_m : word[6];
sig_n : word[6];
sig_gcd : word[6];

um_euclid : mod_euclid(sig_m, sig_n, sig_gcd);

INIT
sig_m = 0d_0;
TRANS
next(sig_m) =
   case
      1 : sig_m;
   esac

INIT
sig_n = 0d_0;
TRANS
next(sig_n) =
   case
      1 : sig_n;
   esac


MODULE mod_euclid(inp_m_in, inp_n_in, out_gcd)

   .
   .   - module body
   .
```

Figure 4.1: NuSMV program structure

4.1.3 Variables

A state of the model is an assignment of values to a set of state variables. Since the language is dedicated for description of Finite State Machines, only finite data types are allowed - boolean, integer, enumeration, word[*] and fixed array. During representation of a model created by the tool three of those types are used:

- Boolean, which compromises two integer numbers 0 and 1 (their symbolic equivalence are FALSE and TRUE).
- Enumeration that is specified by a full enumeration of all possible values for the particular enumeration type.
- Word, which models arrays of bits, allowing bitwise logical and arithmetic operations. This type is distinguishable by its width.

Example of variable declarations for all those types may be found on the figure 4.2.

```
//enumeration
ctl_euclid :  {st_s0, st_s1, st_s2};

// word[*]
reg_done : word[1];
reg_factor : word[6];

// boolean
sfg_init : boolean;
```

Figure 4.2: Declaration of variables.

NuSMV does not provide implicit conversion between those types, however there is a possibility for explicit conversion between boolean and word[1]. The signatures of these conversion operators are presented on the figure 4.3.

```
bool : word[1] -> boolean
word1 : boolean -> word[1]
```

Figure 4.3: Signatures of conversion operators.

4.1.4 Module instantiation

A state variable may be declared as an instance of a module. In this case name of the variable is followed by the module type specifier and if required list of actual parameters. An actual parameter can be any simple expression. The rules for passing of parameters are similar to the call by reference scheme. If the formal parameter appears in an expression within instantiated module it is replaced by the actual parameter. An example for this construct is presented on the figure 4.4. An instance of a module has its own separate namespace.

```
// signature
variable_identifier : module_type_specifier(parameter_list);

// example
um_euclid : mod_euclid(sig_m, sig_n, sig_gcd);
```

Figure 4.4: Module instantiation.

4.1.5 Expressions

The language provides wide range of operators that may be used in expressions. All basic logical, relational, arithmetic operators are all present, together with shift, bit selection and concatenation operators. Two special kinds of operators available in the NuSMV language: case and next, will be discussed in more details.

Syntax of case expression is presented in the figure 4.5 and an example of the usage on the figure 4.6. Value returned by the case expression is equal to the value of the first expression on the right hand side of ':' for which corresponding left hand side expression evaluates to TRUE. The type of the left hand side expression must be boolean.

```
case
  left_expression_1 : right_expression_1 ;
  left_expression_2 : right_expression_2 ;
                    …
  left_expression_n : right_expression_n ;
esac
```

Figure 4.5: Signature of case operator.

Basic next expression refers to next state variables. For example if a variable *v* is a state variable then *next(v)* refers to value of *v* in the next state.

4.1.6 INIT and TRANS constraints

The set of initial states of the system is specified by the use of INIT keyword. An example of usage may be observed on the figure 4.6. Expression used in this constraint cannot contain the next operator and has to be of boolean type.

TRANS constraint specifies transition relation of the model - a set of current state and next state pairs. The syntax for this constraint is presented on the figure 4.6. Expression has to be of boolean type. In case of appearance of more than one TRANS constraint in the module declaration, the transition relation is a conjunction of all those constraints.

```
// signatures
INIT simle_expression ;
TRANS next_expression ;

// example
INIT
sfg_shiftm = FALSE;
TRANS
next(sfg_shiftm) =
   case
      ctl_euclid = st_s0 & (0dl_1) > 0dl_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0dl_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0dl_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0dl_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0dl_0 : TRUE;
      ctl_euclid = st_s1 & (0dl_1) > 0dl_0 : TRUE;
      ctl_euclid = st_s2 & (0dl_1) > 0dl_0 : FALSE;
      1 : FALSE;
   esac
```

Figure 4.6: INIT and TRANS constraints.

4.1.7 Rules for assignment

The way model evolves through time is determined by a system of assignments. In order to ensure that there is a unique solution for the set of assignments several syntactic rules are introduced. As specified in the [5], the restriction rules are:

- The single assignment rule – each variable may be assigned only once.
- The circular dependency rule – a set of equations must not have "cycles", in its dependency graph, not broken by delays.

These rules guarantee that there is at least one solution. It is possible that multiple solutions may occur. This may be the case when unassigned variables appear in the system.

4.1.8 Specifications

The language allows check of the system against specifications expressed in temporal logics like Computational Tree Logic (CTL), Linear Temporal Logic (LTL) extended with Pascal Operators, and Property Specification Language (PSL) that includes CTL and LTL with Sequential Extended Regular Expressions (SERE), a variant of classical regular expressions. Specifications are placed inside a module, they refer to. An example of specification, used in verification of a model created using the tool described in the chapter 3, is presented in the figure 4.7.

```
CTLSPEC AG (sig_s1 = 0d6_0 & sig_s2 = 0d6_0 ->
               sig_s3 = 0d6_0 | sig_s3 = 0d6_0)
CTLSPEC AG (sig_s1 = 0d6_0 & sig_s2 = 0d6_1 ->
               sig_s3 = 0d6_0 | sig_s3 = 0d6_1)
   ...
CTLSPEC AG (sig_s1 = 0d6_63 & sig_s2 = 0d6_63 ->
               sig_s3 = 0d6_0 | sig_s3 = 0d6_63)
```

Figure 4.7: An example of CTL specification.

4.2 Translating hardware models to NuSMV

The tool presented in the chapter 3, provides functionality for creation of NuSMV program that correspond to the currently developed model. In this section the mechanism for creation of this program will be explained.

4.2.1 Variable declarations

Each module from the tool structure is modeled by one module in NuSMV program. Following variables are declared to model a system in terms of NuSMV:

- For every register and signal of the module there is a corresponding state variable declared of the type *word* with length equal to bit number of this register or signal.

- Variables that are passed as parameters from the parent module are used to model input and output variables, so there is no need to declare those inside the module. It is enough for those variables to appear in the parameter list.

- For every sub-module there is generated corresponding instance of sub-module in NuSMV model with the same parameters list.

- For every signal flow graph there is declared a boolean variable, which states whether given signal flow graph is invoked during particular cycle.

- For the controller there is created variable of enumerated type for which the set of possible values contains all controller states.

In order to improve readability of the program and to avoid clash with some NuSMV keyword every variable is prefixed with shortcut for its type: *inp* – input, *out* – output, *sig* – signal, *reg* – register, *sfg* – signal flow graph, *ctl* – controller state. Prefixing is also used for values of enumerated type used by controller variable: *st* – state. An example of variable declarations for NuSMV model is presented on the figure 4.8:

```
VAR

// controller
ctl_euclid :  {st_s0, st_s1, st_s2};

// registers
reg_m : word[6];
reg_n : word[6];
reg_done : word[1];
reg_factor : word[6];

// signal flow graphs
sfg_init : boolean;
sfg_shiftm : boolean;
sfg_shiftn : boolean;
sfg_reduce : boolean;
sfg_shiftf : boolean;
sfg_outidle : boolean;
sfg_complete : boolean;
```

Figure 4.8: Declaration of variables.

4.2.2 Transition declarations

The way, system evolves through time, is modeled by transitions created according to the following pattern:

- Boolean variables which correspond to signal flow graphs: every transition of the model is treated as a separate case, boolean variable is assigned true value if during this particular transition given signal flow graph is invoked, otherwise it is set to false. Transitions are distinguished on the basis of the state of the controller variable and order of appearance together with transition condition. An example may be observed on the figure 4.9.

```
INIT
sfg_shiftm = FALSE;
TRANS
next(sfg_shiftm) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac
```

Figure 4.9: An example definition of signal flow graph variable transition.

- Inputs: there are no transitions because those variables may be used only as right side elements in expressions.

- Signals, registers and outputs: for every signal flow graph that performs calculation of the given variable there is a separate case in transition declaration. Figure 4.10 presents an example of such a definition.

```
INIT
reg_m = 0d6_0;
TRANS
next(reg_m) =
   case
      next(sfg_init)    = TRUE & (0dl_1) > 0dl_0 : next(inp_m_in);
      next(sfg_shiftm) = TRUE & (0dl_1) > 0dl_0 :
                          ((0d2_0 :: (reg_m)) >> ((0dl_1)))[5:0];
      next(sfg_reduce) = TRUE & ((wordl(reg_m >= reg_n))) > 0dl_0 :
                          reg_m - reg_n;
      next(sfg_reduce) = TRUE & (0dl_1) > 0dl_0 : reg_m;
      1 : reg_m;
   esac
```

Figure 4.10: An example definition of register variable transition.

- Controller: similar to signal flow graph boolean variables, each transition is treated as a separate case. Transitions are distinguished based on the current state of the controller, together with order of appearance and condition. Assigned value in this variable is next state for the controller if the given transition is supposed to be taken. Transitions for a sample controller are defined on the figure 4.11.

```
INIT
ctl_euclid = st_s0;
TRANS
next(ctl_euclid) =
   case
      ctl_euclid = st_s0 & (0dl_1) > 0dl_0 : st_s1;
      ctl_euclid = st_s1 & (reg_done) > 0dl_0 : st_s2;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0dl_0 : st_s1;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0dl_0 : st_s1;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0dl_0 : st_s1;
      ctl_euclid = st_s1 & (0dl_1) > 0dl_0 : st_s1;
      ctl_euclid = st_s2 & (0dl_1) > 0dl_0 : st_s2;
      1 : ctl_euclid;
   esac
```

Figure 4.11: An example definition of controller variable transition.

4.2.3 Signal vs. register

The difference in modeling signals and registers lies in the use of the given element on the right side of expressions. According to GEZEL language specification, a signal holds only immediate value, within a single clock cycle. On the other hand NuSMV variables have two states: current and next state. In order to model signal behavior properly, so there is no storage of value over multiple cycles possible we use next state of variable not only during assignment but also in case when element is used as right side element of expressions. In GEZEL language register has two values, current on the output and next on the input, this is behavior is exactly modeled by the typical use of NuSMV variable, which is using current state in the calculation if element appears on the right side.

4.2.4 Differences between GEZEL and NuSMV operators

It is obvious that in order to model GEZEL program properly in terms of NuSMV, expressions from both models need to be resolved identically. Analysis of all GEZEL operators and comparison to the corresponding one provided by NuSMV was necessary. Several differences were traced: difference in domain of accepted values, order of precedence and finally it was spotted that not for every GEZEL operator there is a parallel version in NuSMV at all.

All GEZEL operators allows use of variables of different bit length as operands, conversion is done implicitly. Unfortunately this is not the case with NuSMV operators. Type which is used in modeling variables (word[*]), cannot by implicitly converted to any other type, neither it can be converted to word with different bit length. Conversion must be done explicitly by selection of bit arrays and concatenation. For example, to model GEZEL expression presented on the figure 4.12, where *a* is 4 bit, *b* is 6 bit and *c* is 2 bit signal, NuSMV expression may need several additional bit selection and concatenations with zero vector operations.

```
// GEZEL expression
sig a : word[4];
sig b : word[6]
sig c : word[2];

a = b + c;

// NuSMV representation
a = ( b + ( ( 0d4_0 ) :: c ) )[3:0];
```

Figure 4.12: Type conversion in expressions.

In order to add *b* to *c*, *c* has to be converted to 6 bit variable. It is done by concatenation with 4 bit zero constant (the same mechanism used by GEZEL implicitly, bit length of operation result is equal to the greatest bit length of operands). Finally resulting 6 bit

variable has to be cut to fit variable *a*, for this purpose 4 less significant bits are selected (the same effect like in GEZEL language).

Order of precedence for NuSMV is not exactly the same like for GEZEL. However the structure that stores expressions in MAAOHA tool preserves order of GEZEL expressions. By use of brackets on every operator it is possible to force desired order of precedence in the created NuSMV program, which is actually what the tool does.

4.2.5 Ternary conditional operator

There is an exception in modeling transitions when ternary conditional operator appears in a signal flow graph. NuSMV does not support this operator so it has to be implemented explicitly. For each such an operator transition is divided into two parts as presented on the figure 4.13.

```
// GEZEL ternary conditional
reg_m >= reg_n ? reg_m - reg_n : reg_m;

// NuSMV representation
next(sfg_reduce) = TRUE & ((wordl(reg_m >= reg_n))) > 0dl_0 :
                   reg_m - reg_n;
next(sfg_reduce) = TRUE & (0dl_1) > 0dl_0 : reg_m;
```

Figure 4.13: The ternary conditional representation.

First part models case when condition appears to be met, second part if does not. This way we obtain two cases for a signal flow graph and model behaves exactly like parallel GEZEL model. Nested ternary conditional is implemented in the similar way, simply for each ternary conditional operator there is created one additional case in the way it was presented.

4.2.6 Lookup table operator

Another type of operator that is not present in the NuSMV language and needs some special representation is lookup table. In this case, every single occurrence of the operator results in division of an expression into as many cases as there is elements in the lookup table declared, one case for one possible index. Each such a case is constructed in the following way: to the condition part of the transition there is added comparison, whether expression used inside lookup table operator matches currently considered index, in the expression part the whole expression is written with the lookup table operator replaced by value from the lookup table of specified index. An example of conversion for this operator is presented on the figure 4.14. Exact algorithm for expression conversion, where ternary conditional and lookup table operators are discussed in more details, is described in the section 4.2.7.

```
// GEZEL code
sig value : word[2];
sig index : word[2];
lookup lt : ns(2) = {3, 1, 3, 2};

sfg assign
{
   value = lt(index);
}

// NuSMV representation of sfg assign
INIT
sig_value = 0d2_0;
TRANS
next(sig_value) =
  case
    next(sfg_assign) = TRUE &
      (wordl(next(sig_index) = ((0dl_0) :: 0dl_0))) > 0dl_0
      : 0d2_3;
    next(sfg_assign) = TRUE &
      (wordl(next(sig_index) = ((0dl_0) :: 0dl_1))) > 0dl_0
      : 0dl_0 :: (0dl_1);
    next(sfg_assign) = TRUE &
      (wordl(next(sig_index) = 0d2_2)) > 0dl_0 : 0d2_3;
    next(sfg_assign) = TRUE &
      (wordl(next(sig_index) = 0d2_3)) > 0dl_0 : 0d2_2;
    1 : sig_value;
  esac
```

Figure 4.14: Representation of the lookup table operator.

4.2.7 Detailed description of conversion algorithm

The algorithm for generation of the NuSMV source from a hardware model created using the tool, proceeds as follows. For each module from the model there is a corresponding NuSMV module created. In case the module is the root of the system, parallel NuSMV module is also set to be the root, and therefore named *main*, otherwise original name is preserved, prefixed by *mod_*. Next, the module is checked for existence of inputs and outputs inside its datapath. If these are present, the list of parameters is generated, enclosed by brackets. Each of the parameters is prefixed with its type indicator *inp_* or *out_*. An example presented on the figure 4.15.

```
MODULE mod_modulel (inp_inputl, inp_input2, out_outputl)
```

Figure 4.15: Heading of the module declaration.

Next, the declaration part of the module is generated. Keyword *var* is written to the NuSMV source file. The variable which holds controller state is generated as an enumerated type with a set of possible values equal to the set of controller states. Each

state name is prefixed with *st_*. An example of this declaration may be observed on the figure 4.16.

```
ctl_controller :  {st_statel, st_state2, st_state3};
```

Figure 4.16: The controller variable declaration.

Next, the declaration of registers and signals follows. Prefixing of variable name with type indicator is applied. The variables used to model those elements are declared as word type with number of bits equal to the bit length from hardware model. A sample declaration of a signal and a register is presented on the figure 4.17.

```
reg_registerl : word[registerl_bitlength];
sig_signall : word[signall_bitlength];
```

Figure 4.17: Declaration of signal and register variables.

After that, all sub-modules used inside the datapath of the model are declared. Name of the module prefixed with *um_*, followed by ':', type of the module and list of signals used as parameters enclosed in brackets is used in the declaration. An example of module instantiation may be seen on the figure 4.18.

```
um_modulename : mod_moduletype (sig_signall, sig_signal2);
```

Figure 4.18: Instantiation of the module.

Finally, boolean variables that indicate which signal flow graph is launched during particular cycle are declared. For each signal flow graph there is a separate variable with name of the signal flow graph prefixed with *sfg_* and of boolean type, see the figure 4.19.

```
sfg_signalflowgraph : boolean;
```

Figure 4.19: Declaration of the signal flow graph variable.

When the declaration part is complete for the module, program proceeds to the part where transitions are defined. First, transition is defined for the controller variable. There are several cases, each dealing with a separate transition, which is distinguished based on the state of the controller variable (associated with current state of the controller) and values of the registers (condition).

At the beginning initial value for controller variable is specified. For this purpose initial state for controller is used, see the figure 4.20.

```
INIT ctl_controller = st_initial_state;
```

Figure 4.20: The initialization of the controller variable.

Next, transition heading is declared. Each of the possible transitions is declared as a separate case inside *case – esac* structure, and is created as follows. For each state st_state_i of the controller, every possible transition $tr\_transition_{i,j}$ that starts in the state st_state_i, is treated in a separate case, where condition is constructed on the basis of the current controller state (it is checked whether controller state is equal to $state_i$) and transition specific condition, while assigned value is equal to the next state value for $tr\_transition_{i,j}$. Generation of the exact form of expression used in transition condition will be discussed later in this section. The form of a single transition case is presented on the figure 4.21.

```
ctl_controller = st_statel && transition_condition : st_nextstate
```

Figure 4.21: A single transition case for the controller variable transition.

After all states, together with contained transitions are processed, default case, where controller variable does not change value, is added. Finally, controller variable transition is closed with *esac* keyword, and obtained complete source is of the form as presented on the figure 4.22.

```
INIT
ctl_controller = st_initial_state;
TRANS
next(ctl_controller) =
   case
      ctl_controller = st_initial_state & condition1 : st_statel;
      ctl_controller = st_initial_state & condition2 : st_statel;
      ctl_controller = st_statel & condition3 : st_statel;
      1 : ctl_controller;
   esac
```

Figure 4.22: A complete transition for the controller variable transition.

Next, transitions for registers, signals and outputs variables are defined. For ever such an element following definition is generated. First, if it is a register or signal that is processed, the initial value of the variable is set to zero, presented on the figure 4.23.

```
INIT reg_register1 = 0d8_0;
```

Figure 4.22: Initialization of variables.

For each signal flow graph sfg_i, where assignment of the variable takes place, there is a separate case or cases created in the transition definition. When an expression from the sfg_i that calculates the variable, does not contain any ternary conditional or lookup operators, the condition part of the transition checks whether sfg_i is executed during the particular cycle (whether sfg_ sfg_i is set to *true*, declaration of signal flow graph variables will be explained later in this section) and the assignment part is simply the expression converted using the expression conversion algorithm presented in the section 4.2.8. Otherwise, ternary conditional and lookup operators need to be expressed in terms of NuSMV, and it is done as follows:

1) Every ternary conditional operator (a ? b : c) is expressed in the form of two cases containing simple expressions. In the first case condition *a* from the operator is added to the case condition and expression is *b*. In the second case condition remains unchanged and expression is equal to *c*. An example presented on the figure 4.23 illustrates the flow of the algorithm:

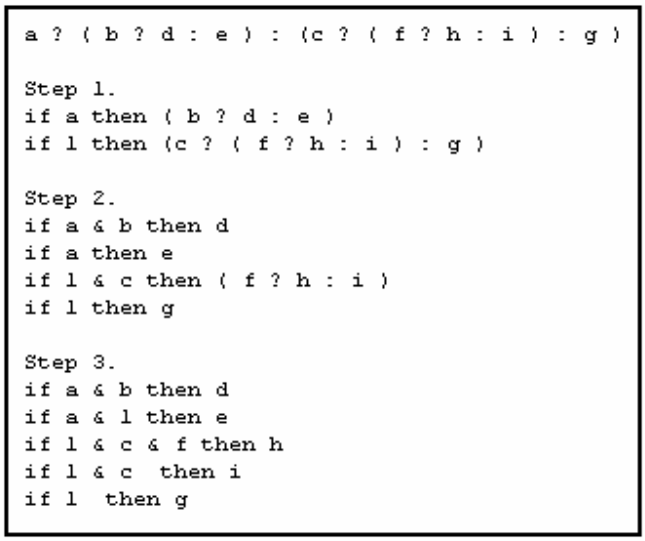

Figure 4.23: An example conversion for the ternary conditional operator.

2) For each obtained case, all lookup table operations are replaced by appropriate constant value, if lookup table index is constant, or again divided into all possible cases, if the index is defined by variable. The figure 4.24 presents an example of such a lookup operator representation.

```
lt = {a ,b}
exp = lt(v) + lt(x)

Step 1.
if v = 1 then exp = a + lt(x)
if v = 2 then exp = b + lt(x)

Step2.
if v = 1 & x = 1 then exp = a + a
if v = 1 & x = 2 then exp = a + b
if v = 2 & x = 1 then exp = b + a
if v = 2 & x = 2 then exp = b + b
```

Figure 4.24: An example conversion for the lookup operator.

3) Finally each possible transition case is written inside the *case – esac* structure.

At the end of transition declaration for this variable, default value is added, where variable preserves current state and the whole declaration is closed. An example of generated transition definition for register would look like it is presented on the figure 4.25.

```
lookup ltab {l1,l2};
sfg sfg1 { reg1 = 0; }
sfg sfg2 { reg1 = reg2 > 0 ? 0 : 1; }
sfg sfg3 { reg1 = reg2 > 0 ? ltab(reg2) : ltab(0); }


INIT
reg_reg1 = 0d8_0;
TRANS
next(reg_reg1) =
   case
      next(sfg_sfg1) = TRUE : 0d8_0;
      next(sfg_sfg2) = TRUE & (reg2 > 0) : 0d8_0;
      next(sfg_sfg2) = TRUE : 0d8_1;
      next(sfg_sfg3) = TRUE & (reg2 > 0) & (reg2 = 0) : 0d8_11;
      next(sfg_sfg3) = TRUE & (reg2 > 0) & (reg2 = 1) : 0d8_12;
      next(sfg_sfg3) = TRUE : 0d8_11;
      1 : reg_reg1;
   esac
```

Figure 4.25: An example of the register variable transition.

When for all signals, registers and outputs transitions are defined, program proceeds to generation of transitions for variables associated with signal flow graphs. For each such a variable following procedure is applied. First, the variable is initialized to *false*. Next, the transition is defined. Similar to the controller variable definition, each transition from the hardware model is treated in its own separate case. For each state st_state_i of the controller, every transition $tr\_transition_{i,j}$ that starts in st_state_i is represented in the separate case, where condition consists of verification if the controller is in the state

st_state$_i$ and the transition specific condition for *tr_transition*$_{i,j}$. If *tr_transition*$_{i,j}$ contains signal flow graph associated with the variable that is being processed assigned value is equal *true* otherwise *false*. Finally, default value is added and *case – esac* structure is closed. An example of the complete transition for the variable associated with the signal flow graph looks as presented on the figure 4.26.

```
INIT
sfg_sfg1 = FALSE;
TRANS
next(sfg_sfg1) =
   case
      ctl_controller = st_initial_state & condition1 : FALSE;
      ctl_controller = st_initial_state & condition2 : TRUE;
      ctl_controller = st_state1 & condition3 : TRUE;
      1 : FALSE
   esac
```

Figure 4.26: An example of the signal flow graph variable transition.

4.2.8 Expression conversion

Expressions in the model, from the tool presented in the chapter 3, are stored in a form of tree structure, where branches are operators, that have one to three operands, and leafs are terminals – constants and variables. Every element of this structure has its type specified, that says whether it is addition, multiplication, lookup table operator, constant etc. Algorithm that generates expression for NuSMV starts at the root element and proceeds as follows:

a) If the element is a branch inside the tree structure, i.e. it is an operator that takes some operands, those operands are created first. For each of the operands algorithm is called recursively. Obtained results are converted to match type of the operand with the greatest bit length. It is done by a concatenation with a zero vector, to simulate the GEZEL implicit conversion. Converted results are bind together using the operation specific syntax. If currently processed operator is the root element, one more conversion is needed to make sure that all bits of the obtained result fit the destination variable. In order to do that appropriate number of less significant bits is selected, again like it is done in the GEZEL language. Finally, the complete result is returned.

b) If the element is a leaf, i.e. variable or constant, appropriate NuSMV expression is created. In case it is the root element, conversion is applied to emulate GEZEL implicit conversion, by selecting number of less significant bits equal to the bit length of destination variable.

If destination of the expression is to be used in condition, there is one more operation added, namely result of the expression is check whether is greatest then zero. Some examples of the expression conversion are presented on the figure 4.27

```
GEZEL: a + 1
NuSMV: a + (( 0d15_0 ) :: 0d1_1);

GEZEL: a << b
NuSMV: ((((a))) << ((b)[3:0]))[15:0];

GEZEL: a = 0 | b = 0
NuSMV: ((word1(a = (( 0d15_0 ) :: 0d1_0))) |
       (word1(b = (( 0d15_0 ) :: 0d1_0))));
```

Figure 4.27: Examples of expressions used in assignment.

```
GEZEL: a
NuSMV: (a) > 0d1_0

GEZEL: a[0] & b[0]
NuSMV: (a[0:0] & b[0:0]) > 0d1_0

GEZEL: a > b
NuSMV: ((word1(a > b))) > 0d1_0
```

Figure 4.28: Examples of expressions used in condition.

4.3 NuSMV model of Euclidean algorithm

In this section a NuSMV model is presented, generated in order to perform full functional test on the implementation of Euclid algorithm introduced in the section 3.5. The program consists of three modules that correspond to modules from the original hardware description.

```
dp euclid(in  m_in, n_in : ns(16);
          out gcd        : ns(16)) {
  reg m, n               : ns(16);
  reg done               : ns(1);
  reg factor             : ns(16);

  sfg init    { m = m_in; n = n_in; factor = 0;
                $display("cycle=", $cycle, " m=", m_in, " n=", n_in); }
  sfg shiftm  { m = m >> 1; }
  sfg shiftn  { n = n >> 1; }
  sfg reduce  { m = (m >= n) ? m - n : m;
                n = (n >  m) ? n - m : n; }
  sfg shiftf  { factor = factor + 1; }
  sfg outidle { gcd = 0; done = ((m == 0) | (n == 0)); }
  sfg complete{ gcd = ((m > n) ? m : n) << factor;
                $display("cycle=", $cycle, " gcd=", gcd); }
}
```

```
fsm euclid_ctl(euclid) {
  initial s0;
  state s1, s2;

  @s0 (init, outidle) -> s1;
  @s1 if (done)               then (complete)          -> s2;
      else if ( m[0] &  n[0]) then (reduce, outidle)   -> s1;
      else if ( m[0] & ~n[0]) then (shiftn, outidle)   -> s1;
      else if (~m[0] &  n[0]) then (shiftm, outidle)   -> s1;
                              else (shiftn, shiftm,
                                    shiftf, outidle)   -> s1;
  @s2 (outidle) -> s2;
}

//--- Testbench
dp test_euclid(out m, n : ns(16)) {
  always {
    m = 2322;
    n = 654;
  }
}

dp euclid_sys {
   sig m, n, gcd : ns(16);
   use euclid(m, n, gcd);
   use test_euclid(m, n);
}

system S {
  euclid_sys;
}
```

Figure 4.29: Euclid algorithm in GEZEL used as a source for NuSMV generation.

4.3.1 The main module

The *mod_euclid_sys* module composes instances of *mod_euclid* and mod_*test_euclid* modules. Signals *sig_m* and *sig_n* pass input vectors from *mod_test_module* to *mod_eculid*, while signal *sig_gcd* serves as an output variable for the calculated greatest common divisor. The source of the main module may be observed on the figure 4.30.

```
MODULE main

VAR

sig_m : word[16];
sig_n : word[16];
sig_gcd : word[16];

um_euclid : mod_euclid(sig_m, sig_n, sig_gcd);
um_test_euclid : mod_test_euclid(sig_m, sig_n);
```

Figure 4.30: NuSMV representation of the euclid_sys module.

4.3.2 The euclid module

The *mod_euclid* module performs GCD calculation of the two input values in the variables *inp_m_in* and *inp_n_in* and writes the result into *out_gcd*.

```
MODULE mod_euclid(inp_m_in, inp_n_in, out_gcd)

VAR

ctl_euclid :  {st_s0, st_s1, st_s2};

reg_m : word[16];
reg_n : word[16];
reg_done : word[1];
reg_factor : word[16];


sfg_init : boolean;
sfg_shiftm : boolean;
sfg_shiftn : boolean;
sfg_reduce : boolean;
sfg_shiftf : boolean;
sfg_outidle : boolean;
sfg_complete : boolean;

INIT
ctl_euclid = st_s0;
TRANS
next(ctl_euclid) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : st_s2;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : st_s1;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : st_s2;
      1 : ctl_euclid;
   esac

INIT
out_gcd = 0d16_0;
TRANS
next(out_gcd) =
   case
      next(sfg_outidle) = TRUE & (0d1_1) > 0d1_0 : 0d15_0 :: (0d1_0);
      next(sfg_complete) = TRUE & ((word1(reg_m > reg_n))) > 0d1_0 :
             ((((reg_m))) << ((reg_factor)[3:0]))[15:0];
      next(sfg_complete) = TRUE & (0d1_1) > 0d1_0 :
             ((((reg_n))) << ((reg_factor)[3:0]))[15:0];
      1 : out_gcd;
   esac

INIT
reg_m = 0d16_0;
TRANS
next(reg_m) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : next(inp_m_in);
```

```
      next(sfg_shiftm) = TRUE & (0d1_1) > 0d1_0 :
             (((reg_m)) >> ((0d1_1)))[15:0];
      next(sfg_reduce) = TRUE & ((word1(reg_m >= reg_n))) > 0d1_0 :
             reg_m - reg_n;
      next(sfg_reduce) = TRUE & (0d1_1) > 0d1_0 : reg_m;
      1 : reg_m;
   esac

INIT
reg_n = 0d16_0;
TRANS
next(reg_n) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : next(inp_n_in);
      next(sfg_shiftn) = TRUE & (0d1_1) > 0d1_0 :
             (((reg_n)) >> ((0d1_1)))[15:0];
      next(sfg_reduce) = TRUE & ((word1(reg_n > reg_m))) > 0d1_0 :
             reg_n - reg_m;
      next(sfg_reduce) = TRUE & (0d1_1) > 0d1_0 : reg_n;
      1 : reg_n;
   esac

INIT
reg_done = 0d1_0;
TRANS
next(reg_done) =
   case
      next(sfg_outidle) = TRUE & (0d1_1) > 0d1_0 : ((word1(reg_m = (( 0d15_0 )
             :: 0d1_0))) | (word1(reg_n = (( 0d15_0 ) :: 0d1_0))));
      1 : reg_done;
   esac

INIT
reg_factor = 0d16_0;
TRANS
next(reg_factor) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : 0d15_0 :: (0d1_0);
      next(sfg_shiftf) = TRUE & (0d1_1) > 0d1_0 :
             reg_factor + (( 0d15_0 ) :: 0d1_1);
      1 : reg_factor;
   esac

INIT
sfg_init = FALSE;
TRANS
next(sfg_init) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftm = FALSE;
TRANS
next(sfg_shiftm) =
   case
```

```
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftn = FALSE;
TRANS
next(sfg_shiftn) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_reduce = FALSE;
TRANS
next(sfg_reduce) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftf = FALSE;
TRANS
next(sfg_shiftf) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_outidle = FALSE;
TRANS
next(sfg_outidle) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
```

```
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac

INIT
sfg_complete = FALSE;
TRANS
next(sfg_complete) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac
```

Figure 4.31: NuSMV representation of the euclid module.

4.3.3 The test_euclid module

The *mod_test_euclid* module assigns the same values to variables *out_m* and *out_n* during each cycle, generating input vectors for GCD calculation.

```
MODULE mod_test_euclid(out_m, out_n)

VAR

ctl_test_euclid :  {st_s0};



sfg_always : boolean;

INIT
ctl_test_euclid = st_s0;
TRANS
next(ctl_test_euclid) =
   case
      ctl_test_euclid = st_s0 & (0d1_1) > 0d1_0 : st_s0;
      1 : ctl_test_euclid;
   esac

INIT
out_m = 0d16_0;
TRANS
next(out_m) =
   case
      next(sfg_always) = TRUE & (0d1_1) > 0d1_0 : 0d4_0 :: (0d12_2322);
      1 : out_m;
   esac
```

```
INIT
out_n = 0d16_0;
TRANS
next(out_n) =
   case
      next(sfg_always) = TRUE & (0d1_1) > 0d1_0 : 0d6_0 :: (0d10_654);
      1 : out_n;
   esac

INIT
sfg_always = FALSE;
TRANS
next(sfg_always) =
   case
      ctl_test_euclid = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac
```

Figure 4.32: NuSMV representation of the test_euclid module.

4.3.4 CTL specifications

In order to perform verification, CTL specifications that describe expected results for the program output were created and added to the main module. It is checked whether for all possible values of input variables *m* and *n*, value of the signal *gcd* is always equal to zero or to greatest common divisor of those two numbers. Specifications are presented on the figure 4.33.

```
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_0)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
        .
        .
        .
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_31)
```

Figure 4.33: CTL specifications added to the main module.

4.3.5 Verification results

Generated model with specified CTL specifications is ready to be verified by NuSMV. Each single specification is evaluated separately and in case when it turns out to be false, counter example is generated.

Successful verification was performed for the models of Euclidean algorithm with variables up to 5 bit long. The output of verification may be seen on the figure 4.34.

```
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 | sig_gcd = 0d5_0))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 | sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 | sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 | sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 | sig_gcd = 0d5_4))  is true
        .
        .
        .
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 | sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 | sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 | sig_gcd = 0d5_31))  is true
```

Figure 4.34: NuSMV verification output.

5 Design

The main idea during design of the tool was to make implementation as flexible as possible. In order to achieve this, it was decided to divide the tool source into several cooperating but still separate components. There is a clear distinction between which part is responsible for the model structure and which for its view. Also, all parts of the code, responsible for reading from and to the file, follow the same convention. Components are independent of each other, all communication and data exchange is made through the model structure. Moreover, the model structure part is simple and does not contain or depend on any redundant information. Therefore, model created in the tool can be easily used by any external application, simply by importing its internal structure from the XML file. All components and interaction between them is presented on the figure 5.1.

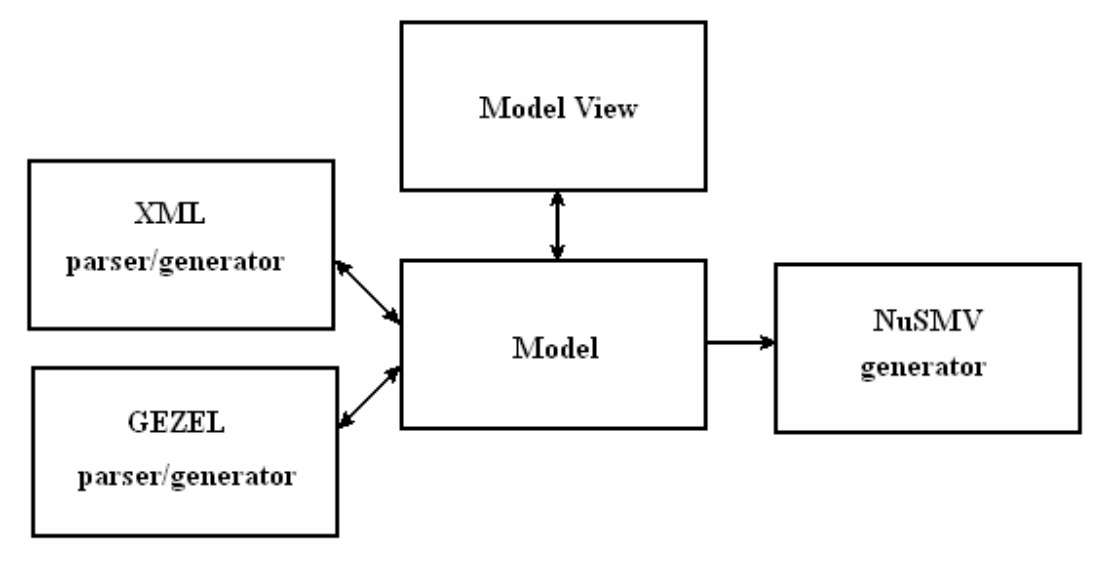

Figure 5.1: Structure of the implementation.

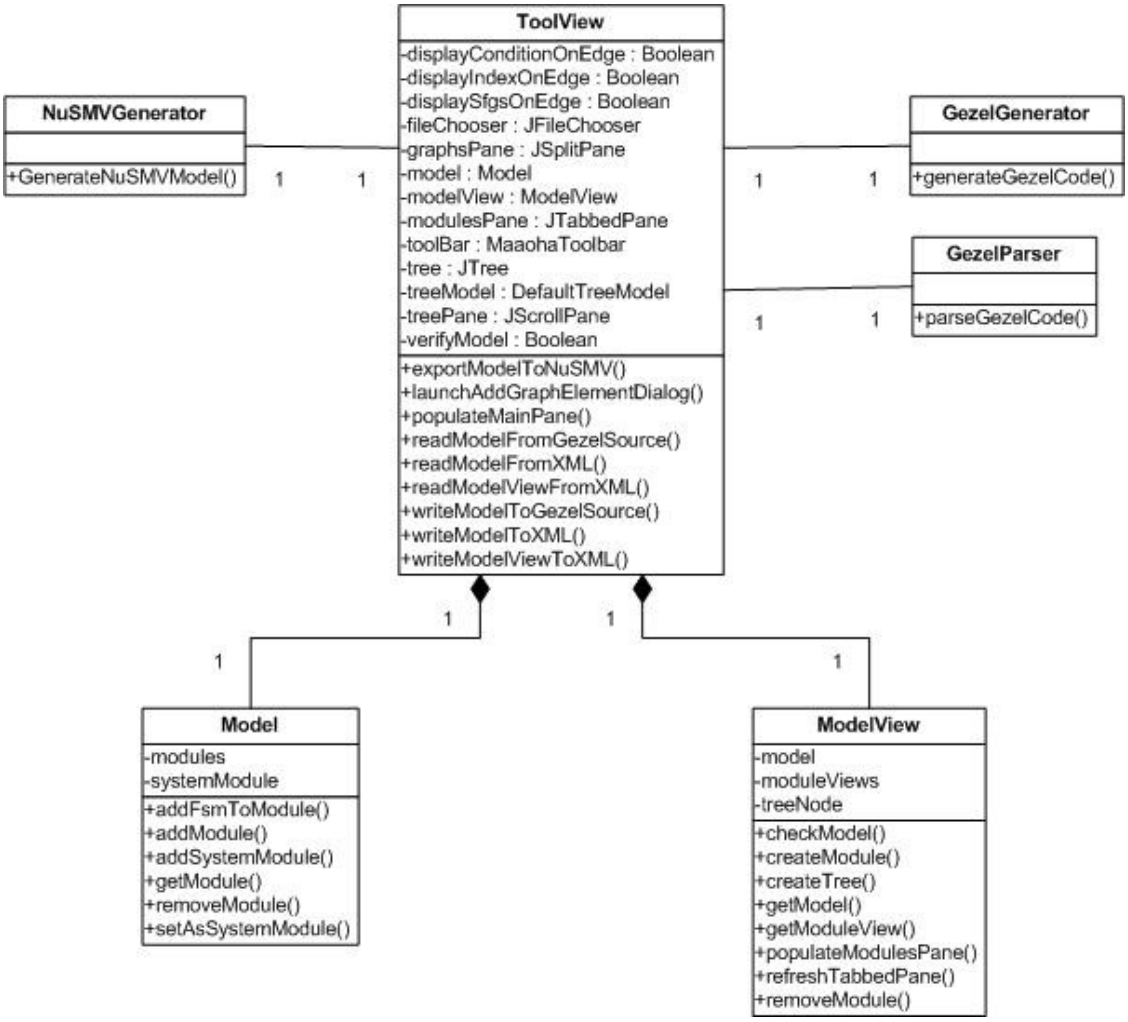

Figure 5.2: UML diagram of the tool structure

5.1 The model component

Representation of the hardware model within the tool is stored in a form of tree structure, created by instances of classes from the model component. The idea behind design of this component is to follow main concepts from GEZEL language. Basically, each single construct type from the language has its own corresponding class in this component, that models behavior of this construct.

5.1.1 Model and modules

The root element for the model component is the Model class. It is a simple class which main purpose is to encapsulate instances of the class that represents modules (a datapath and its controller) - the Module class. The Module class provides functionality for

operations that require treatment of a module as a whole. Otherwise, the call is directed to and resolved in the specific class that represents the datapath or the controller, respectively classes Datapath and Fsm.

5.1.2 The datapath

The Datapath class groups declaration of datapath elements like variables, instances of sub-modules and signal flow graphs. Each of those element types is modeled by its own dedicated class. These classes are named respectively BasicElement, UseModule and Sfg. Instance of the BasicElement class represents a single variable, specifying its name, type of the variable, whether it is input, output, signal or register, and its value type. The UseModule class, which corresponds to a single instance of a sub-module, consists of name and parameter list for this instance and reference to the Module class that represents particular module kind. An instance of the Sfg class models single signal flow graph, it consists of list of instances of Expression class – class that represents single expression. The structure of the Expression class will be discussed in the section 5.2 of this report.

5.1.3 The controller

The Fsm class that represents controller consists of list of instances of the FsmState class, which models each possible state of the controller. The FsmState class holds list of transitions, implemented by the FsmTransition class. The FsmTransition class consists of three parts:

- Condition that is modeled by the same class as expressions.
- List of Sfg instances declared in the datapath of the parent module, where those instances correspond to the signal flow graphs invoked during particular transition.
- A reference to the FsmState that corresponds to the next state of the controller after the transition is taken.

All classes presented in this section except representation of the hardware model, provide some basic functionality for manipulation on the structure of this model, like addition, removal or edition. UML diagram for this component is presented on the figure 5.3.

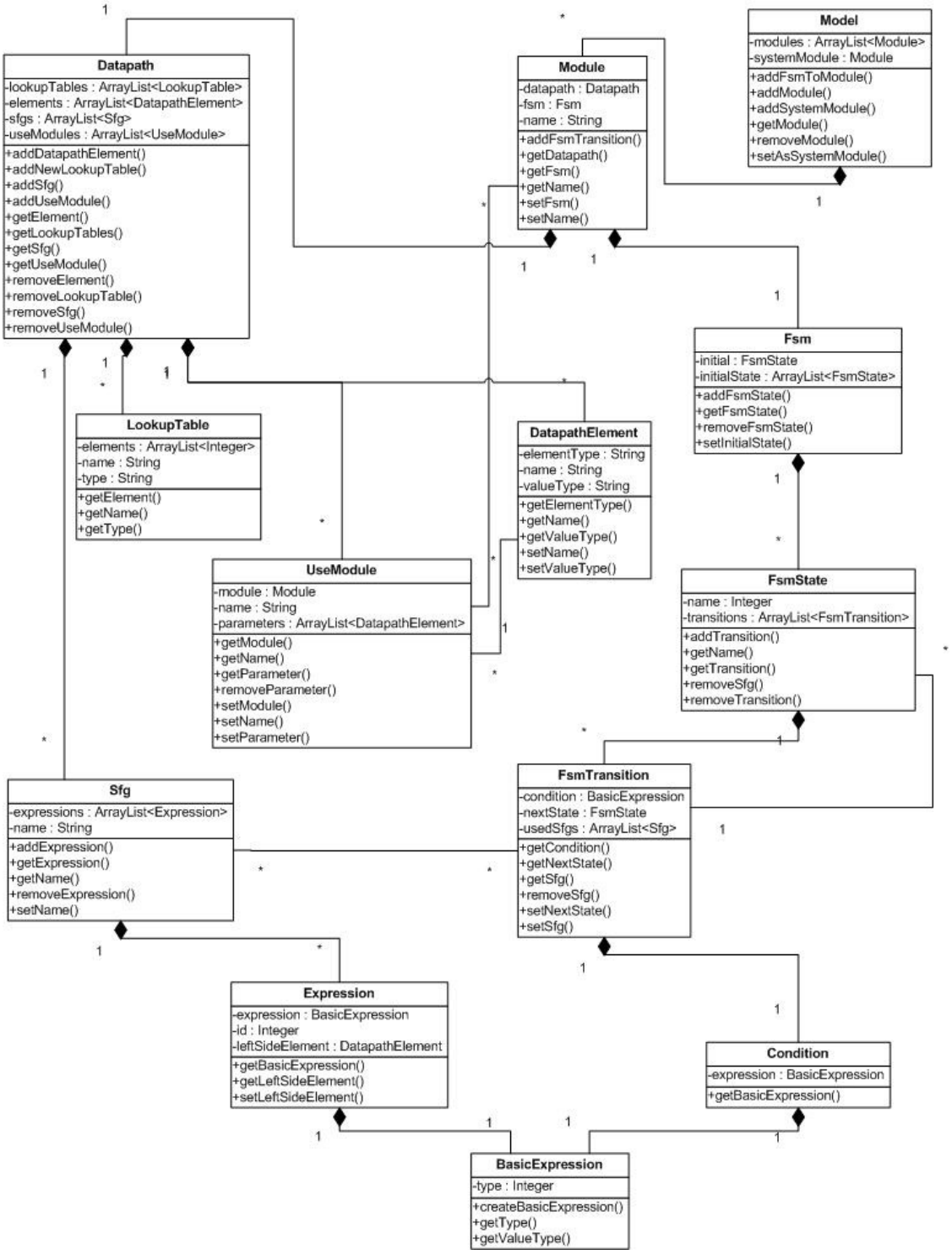

Figure 5.3: UML diagram of the model component.

5.2 Expressions

Representation of the expression structure, used by classes Condtion and Expression, is implemented in the maaoha.model.expression package. The idea, behind creation so many classes for this structure, lies in fact that various operator kinds and operands reveal different properties.

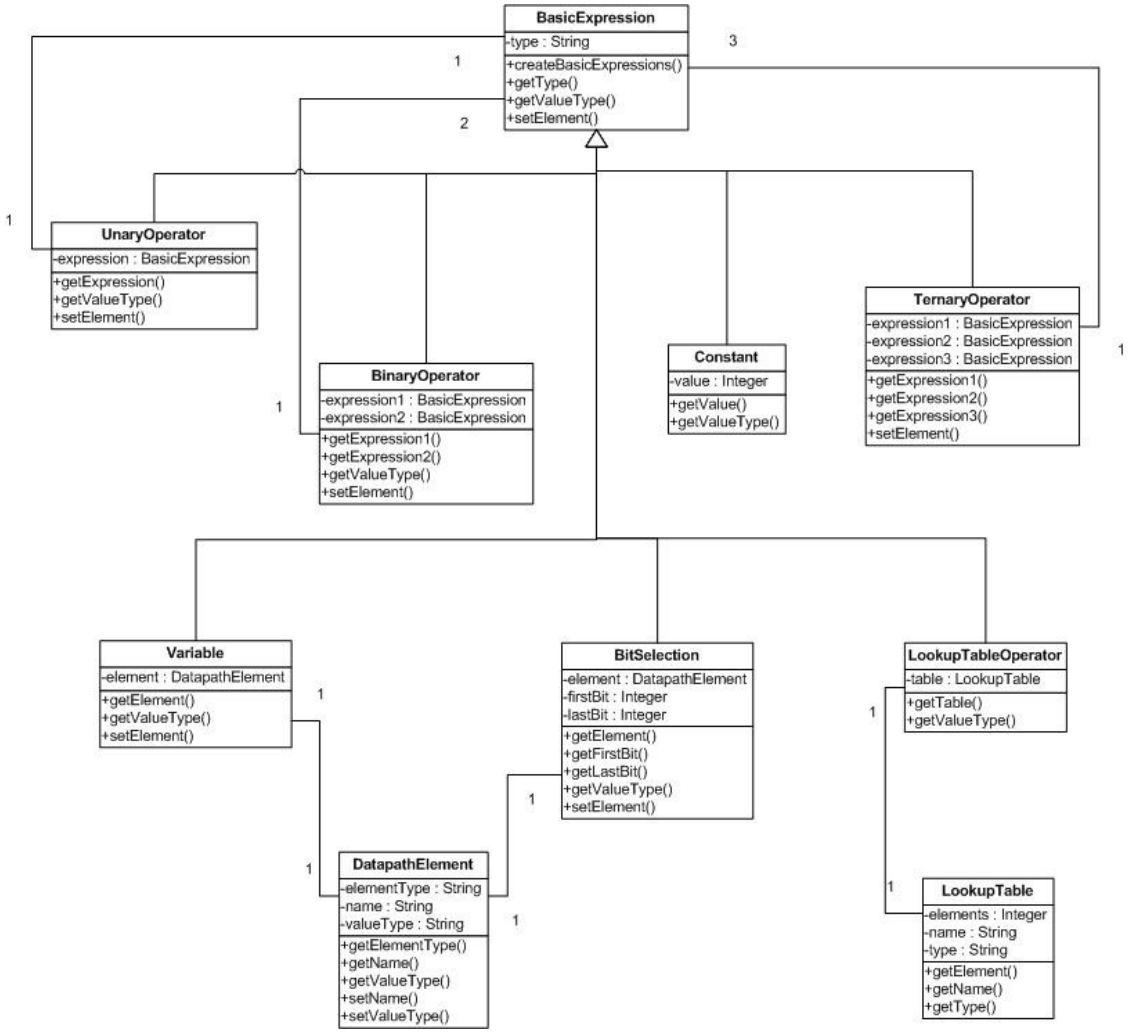

Figure 5.4: UML diagram of the expression structure.

Basically elements which may appear on the right side of expression fall into several categories:

- Unary operators (negation, brackets)
- Binary operators (addition, comparison operators)
- Ternary operators (ternary conditional)
- Lookup tables
- Bit selection
- Variable
- Constant

For each of those categories there is a separate dedicated class that inherits from the BasicExpression class. This inheritance allows nesting of operators and operands, use of different kind of elements inside UnaryOperator, BinaryOperator and TernaryOperator classes. UML diagram of this structure is presented on the figure 5.4.

5.3 The model-view component

All functionality responsible for visualization is separated from the structure that represents hardware model, and implemented in its own component. The component consists of classes structured in the similar way like it is done in the model component. Namely, each class from the model-view component is built on the top of corresponding class from the model component and links this class with its graphical representation in the tool (a JTree node and for optionally a graph cell). The structure of the model-view component together with its connection to the model component may be seen on the figures 5.5 and 5.6.

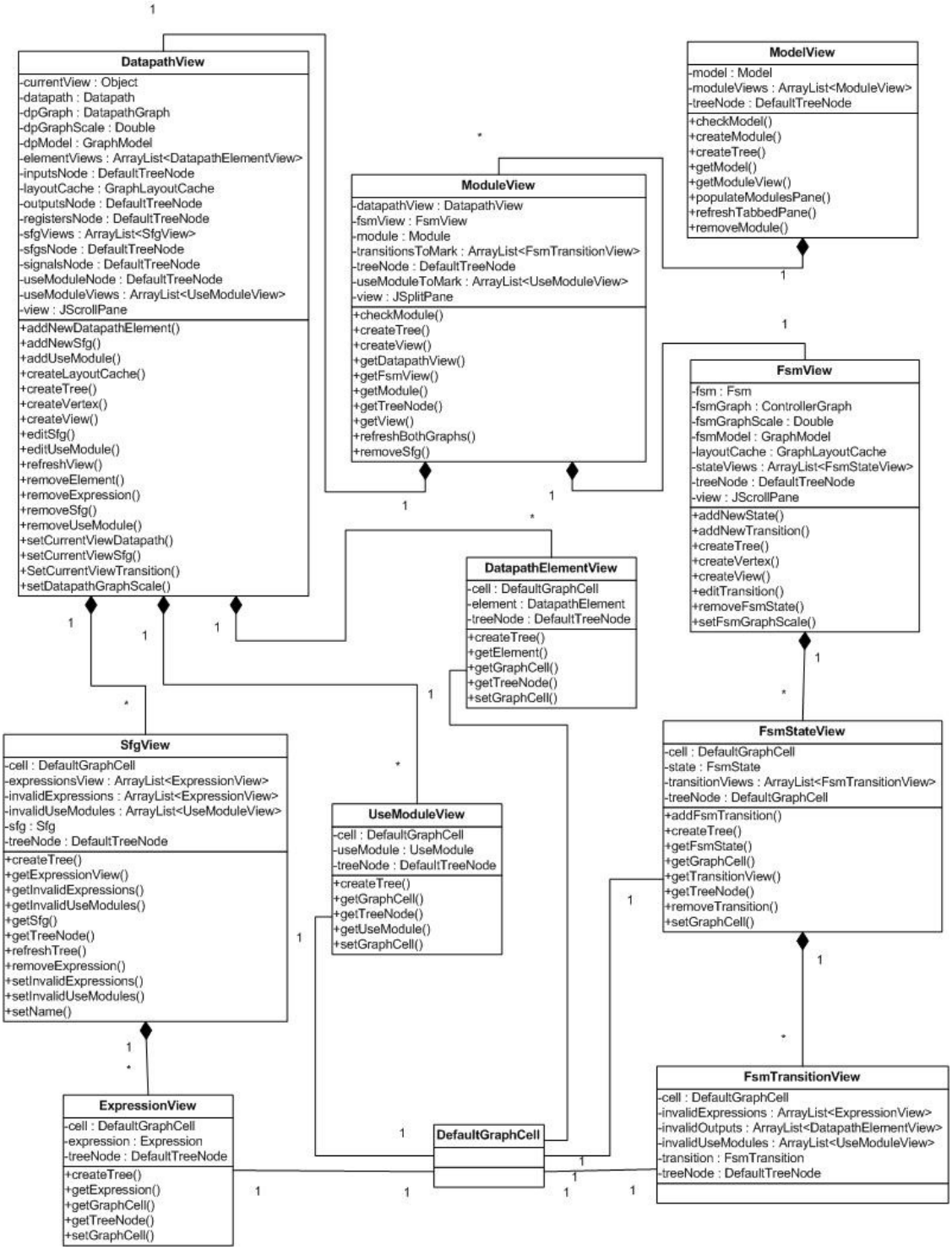

Figure 5.4: UML diagram of the model view component.

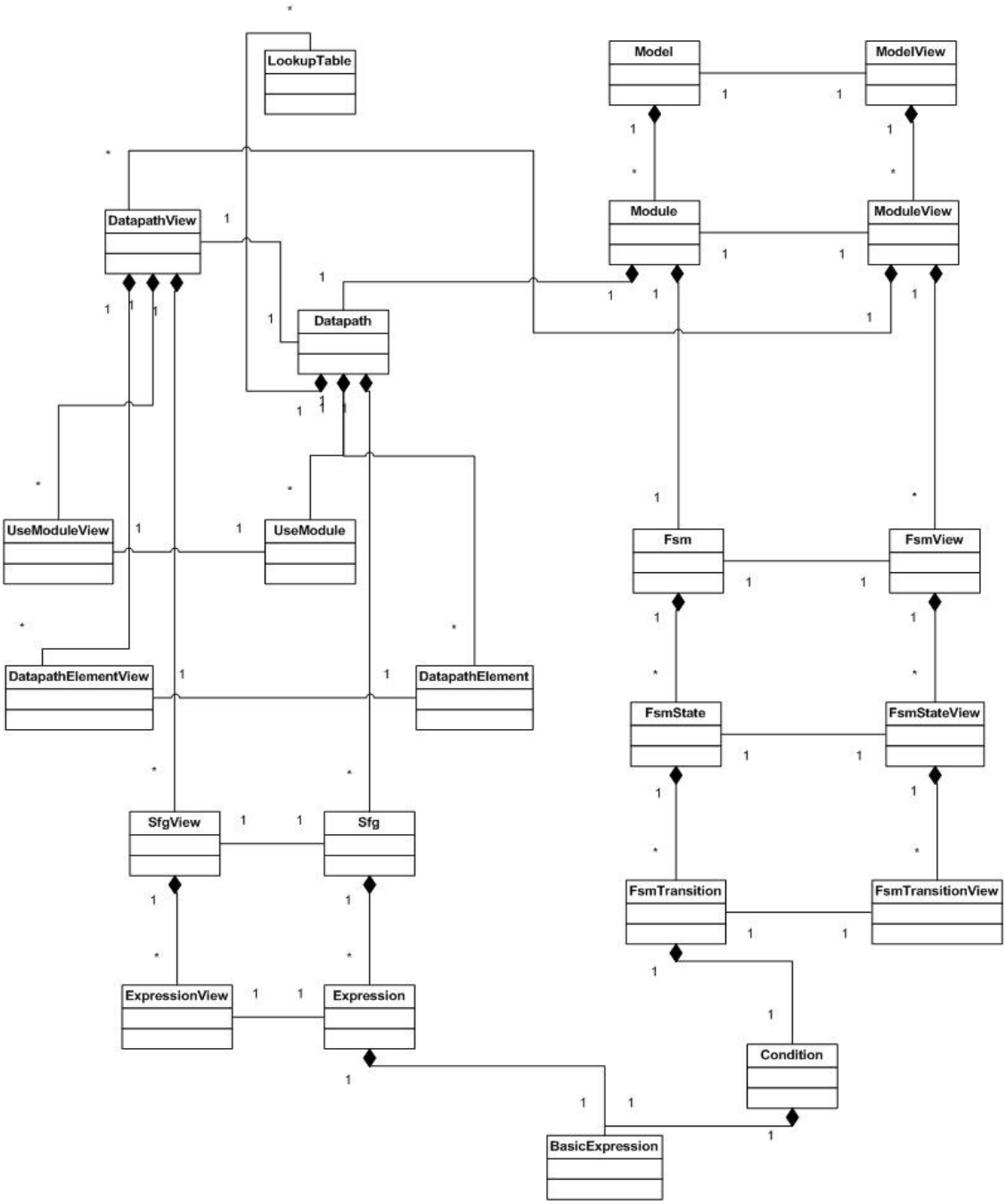

Figure 5.6: UML diagram of the model structure and its connection to the model view.

5.4 The JGraph library

Visualization of the datapath and the controller in the form of graphs is done using JGraph. JGraph is a feature-rich open source graph visualization library written in Java, fully compatible with Swing. Through its programming API provides the means to configure how the graph is displayed and the means to associate a context with those displayed elements.

5.4.1 The JGraph class

The JGraph class is the core of the library. Each graph implemented using the JGraph library requires its own instance of this class. Since it extends JComponent it can be easily use by the application, the same way like any other component. The JGraph class binds together the graph model, graph views and the user interface handling functionality. It provides top-level configuration of the graph, like handling of the mouse events, allowing edition, dragging and resizing of graph elements, setting background.

In the tool implementation, both DatapathView and FsmView use classes derived from this class, respectively DatapathGraph and FsmGraph, with all graph specific settings declared inside.

5.4.2 JGraph Model-View-Controller

JGraph separates model data from its graphical representation. Class responsible for representation of the graph structure is GraphModel, while implementation of a view of the graph is contained in the GraphLayoutCache class. The view is logically built on the top of the graph model and automatically updates in respond to any changes introduced in the model that it is attached to. A graph model can have several views attached, therefore allowing different visual representation of the same graph data.

Use of several different views was initially considered in order to model diversity of views for datapath. Finally, it was chosen rather to show and hide appropriate vertices in one view, then create a separate object for each possible view of the datapath. The reason for that was concern about application performance in a case of greater number of possible views. Storing separate data structure for every such a case and keeping all those views consistent seemed unnecessary burden for the application, especially when use of one view can provide application with all functionality it needs. Initialization of the datapath and the controller graphs is implemented in the methods CreateView of the DatathView and FsmView classes, respectively, and it may be observed on the figure 5.7.

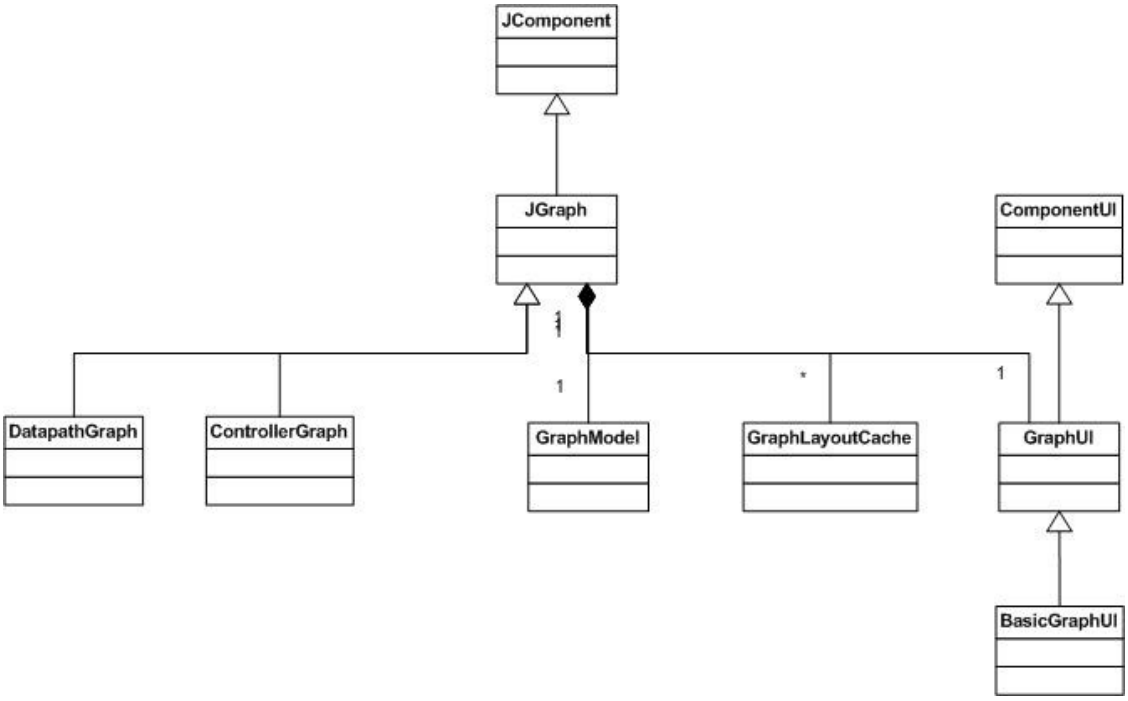

Figure 5.7: JGraph Model-View-Controller.

```
dpModel = new DefaultGraphModel();
layoutCache =
   new GraphLayoutCache(dpModel, new GPCellViewFactory(), true);
dpGraph = new DatapathGraph(dpModel, layoutCache);
```

Figure 5.8: Datapath graph initialization.

5.4.3 The graph model

A graph model, both for the datapath and the controller, is created using the JGraph DefaultGraphModel class. The structure of the model is built up from instances of the following classes: DefaultGraphCell, DefaultEdge and DefaultPort. DefaultGraphCell is the standard implementation of a graph vertex provided by JGraph and it is used as-is in the application. The creation of the vertex is implemented in the method CreateVertex of the DatapathView and the FsmView class. Instances of DefaultEdge represent edges in the model. Each such an object holds two references to DefaultPort objects that indicate source and target of the edge. DefaultPort represents an entity conceptually associated with a vertex. Instances of this class bind an associated vertex object with objects that represent edges connected to this vertex.

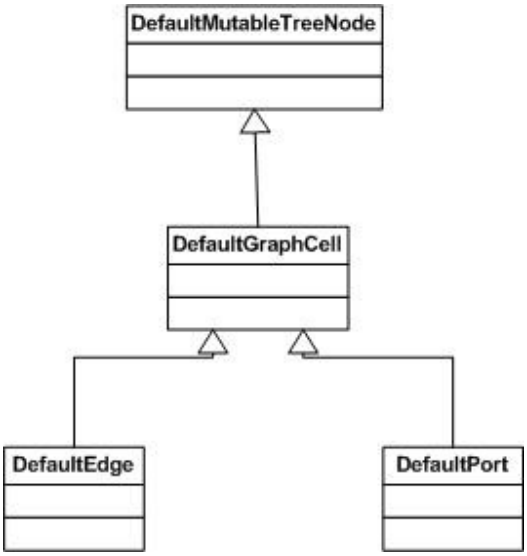

Figure 5.9: UML diagram of GraphModel elements structure.

In the implementation of the tool instances of DefaultGraphCell are encapsulated in dedicated view classes. Those view classes provided communication between the application and the JGraph model structure. Depending on the type of the element that given vertex is supposed to represent, encapsulating class can be: DatapathElementView for inputs, outputs, registers and signals, ExpressionView for expressions, UseModuleView for sub-modules and FsmStateView for states of controller. For controller graph edges are encapsulated in FsmTransitionView because they are associated with transitions while for datapath there is no specific class for encapsulation of edges, since they do not represent any particular structure and are used only for informative purposes to point elements used in expressions and as parameters is sub-module declarations.

5.4.4 Cell views

All graph cells have at least one associated cell view that deals with visual functionality for that cell. Cell view links each graph cell with its renderer, cell editor and cell handle. According to the Swing design, renderers abstract the drawing functionality of components into a single static class instance, so all of those components can share the same common renderer instance, just with different visual attributes - so called flyweight design. Editors associated with graph cells are similar to cell editors for the JTable and the JTree components. They provide so called in-place edition, in case of double-click on a vertex or edge cell. Cell handles provide visual representation that indicates process of moving or resizing of the cell.

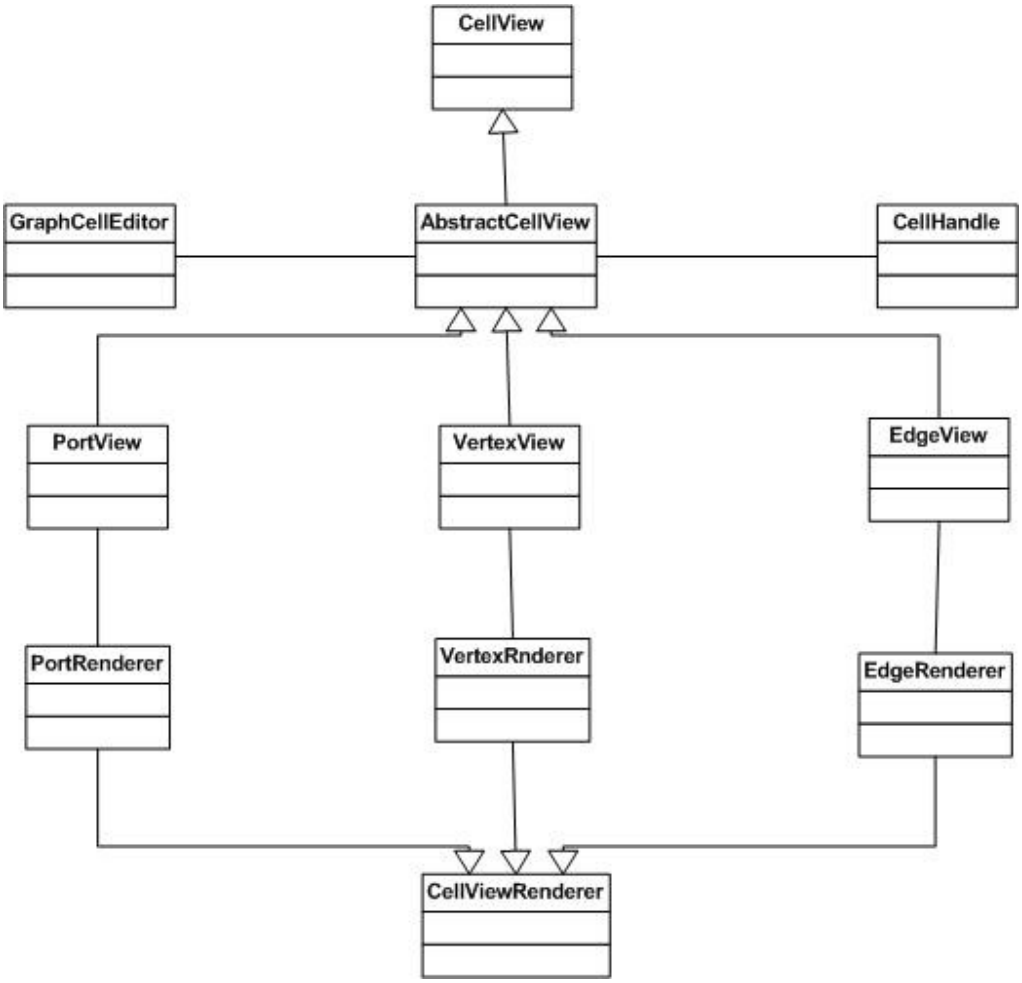

Figure 5.10: The CellView interface.

Almost all graph cells in the application use default constructs for visual representation. There is only one class added that extends VertexView class, namely JGraphEllipseView, which is responsible for drawing ellipse shapes that are being used to represent expressions in the datapath graph.

5.5 Serialization

A model edited in the application can be exported to and imported from an XML files. There are two possibilities for that: serialization of the model logical structures alone or together with all data necessary to re-instantiate the graphical representation. In the first case only object that is written to a file is an instance of the Model class. If full serialization of the project is needed, a Model object is bind together with all necessary data extracted from the view structure, inside the instance of the ModelData, dedicated class for this specific purpose. In the implementation of the tool's main class ToolView there are methods readModelFromXML and writeModelToXML that perform

serialization operations on the model alone, and methods readModelViewFromXML and writeModelViewToXML that do the serialization for the model together with all data necessary to reinstantiate its graphical representation.

Generation of the XML file, both for instances of the Model or the ModelData class, is done using XStream. XStream is a simple library, capable of serialization Java classes in the form of XML file. The library produces clean XML, no information is duplicated, result is easy to read and more compact then native Java serialization. Some examples of generated XML files may be seen on the figures 5.11 and 5.12.

```
<model>
  <modules>
    <module>
      <name>module_name</name>
      <datapath>
        <lookupTables/>
        <elements>
          <element>
            <name>register_name</name>
            <elementType>register</elementType>
            <valueType>ns(8)</valueType>
          </element>
        </elements>
        <sfgs>
          <sfg>
            <name>sfg_name</name>
            <expressions>
              <expression>
                <leftSideElement reference="../../../../../elements/element"/>
                <rightSideElements>
                  <element reference="../../../../../../elements/element"/>
                </rightSideElements>
                <id>0</id>
                <expression class="binaryOperator">
                  <expression1 class="constant">
                    <value>2</value>
                    <type>19</type>
                  </expression1>
                  <expression2 class="variable">
                    <element reference="../../../../../
                          ../../elements/element"/>
                    <type>20</type>
                  </expression2>
                  <type>17</type>
                </expression>
              </expression>
            </expressions>
          </sfg>
        </sfgs>
        <useModules/>
        <parentModule reference="../.."/>
      </datapath>
      <fsm>
        <states>
          <fsmState>
            <name>state_name</name>
            <transitions>
              <fsmTransition>
```

```
              <condition>
                <expression class="constant">
                  <value>1</value>
                  <type>19</type>
                </expression>
              </condition>
              <usedSfgs>
                <sfg reference="../../../../../../../datapath/sfgs/sfg"/>
              </usedSfgs>
              <nextState reference="../../.."/>
            </fsmTransition>
          </transitions>
        </fsmState>
      </states>
    </fsm>
    <parentModel reference="../../.."/>
  </module>
 </modules>
 <ExpressionCounter>0</ExpressionCounter>
</model>
```

Figure 5.11: XML file of a model.

```
<maaoha.view.serialization.ModelData>
  <model>
    <modules>
      <module>
        <name>module_name</name>
        <datapath>
          <lookupTables/>
          <elements>
            <element>
              <name>register_name</name>
              <elementType>register</elementType>
              <valueType>ns(8)</valueType>
            </element>
          </elements>
          <sfgs>
            <sfg>
              <name>sfg_name</name>
              <expressions>
                <expression>
                  <leftSideElement reference="../../../../../
                          elements/element"/>
                  <rightSideElements>
                    <element reference="../../../../../../
                          elements/element"/>
                  </rightSideElements>
                  <id>0</id>
                  <expression class="binaryOperator">
                    <expression1 class="constant">
                      <value>2</value>
                      <type>19</type>
                    </expression1>
                    <expression2 class="variable">
                      <element reference="../../../../../../../
                          elements/element"/>
                      <type>20</type>
                    </expression2>
```

```
                <type>17</type>
              </expression>
            </expression>
          </expressions>
        </sfg>
      </sfgs>
      <useModules/>
      <parentModule reference="../.."/>
    </datapath>
    <fsm>
      <states>
        <fsmState>
          <name>state_name</name>
          <transitions>
            <fsmTransition>
              <condition>
                <expression class="constant">
                  <value>1</value>
                  <type>19</type>
                </expression>
              </condition>
              <usedSfgs>
                <sfg reference="../../../../../../../
                      datapath/sfgs/sfg"/>
              </usedSfgs>
              <nextState reference="../../.."/>
            </fsmTransition>
          </transitions>
        </fsmState>
      </states>
    </fsm>
    <parentModel reference="../../.."/>
  </module>
 </modules>
 <ExpressionCounter>0</ExpressionCounter>
</model>
<modules>
 <maaoha.view.serialization.ModuleData>
  <datapathElements>
   <maaoha.view.serialization.ElementData>
    <object class="element" reference="../../../../../
       model/modules/module/datapath/elements/element"/>
    <bounds class="org.jgraph.graph.
       AttributeMap$SerializableRectangle2D"
       serialization="custom">
      <unserializable-parents/>
      <java.awt.geom.Rectangle2D_-Double>
        <default>
          <height>40.0</height>
          <width>40.0</width>
          <x>414.0</x>
          <y>166.0</y>
        </default>
      </java.awt.geom.Rectangle2D_-Double>
      <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
        <default/>
        <double>414.0</double>
        <double>166.0</double>
        <double>40.0</double>
        <double>40.0</double>
      </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
    </bounds>
   </maaoha.view.serialization.ElementData>
```

```
      </datapathElements>
      <controllerElements>
        <maaoha.view.serialization.ElementData>
          <object class="fsmState" reference="../../../../../
             model/modules/module/fsm/states/fsmState"/>
          <bounds class="org.jgraph.graph.
             AttributeMap$SerializableRectangle2D"
             serialization="custom">
            <unserializable-parents/>
            <java.awt.geom.Rectangle2D_-Double>
              <default>
                <height>50.0</height>
                <width>50.0</width>
                <x>401.0</x>
                <y>93.0</y>
              </default>
            </java.awt.geom.Rectangle2D_-Double>
            <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
              <default/>
              <double>401.0</double>
              <double>93.0</double>
              <double>50.0</double>
              <double>50.0</double>
            </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
          </bounds>
        </maaoha.view.serialization.ElementData>
      </controllerElements>
      <edgeLabelsElements>
        <maaoha.view.serialization.EdgeData>
          <transition reference="../../../../../
             model/modules/module/fsm/states/fsmState/transitions/
             fsmTransition"/>
          <labelPosition class="org.jgraph.graph.
             AttributeMap$SerializablePoint2D" serialization="custom">
            <unserializable-parents/>
            <java.awt.geom.Point2D_-Double>
              <default>
                <x>500.0</x>
                <y>0.0</y>
              </default>
            </java.awt.geom.Point2D_-Double>
            <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
              <default/>
              <double>500.0</double>
              <double>0.0</double>
            </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
          </labelPosition>
        </maaoha.view.serialization.EdgeData>
      </edgeLabelsElements>
    </maaoha.view.serialization.ModuleData>
  </modules>
</maaoha.view.serialization.ModelData>
```

Figure 5.12: XML file of a model together with graphical representation data.

5.6 Parsing of the GEZEL program

The tool provides a possibility to import a hardware model description from a GEZEL source. In order to do that, the application reads program line by line, separates the lines into atomic elements (keywords, variable names and operators), reads the first token and enters the main loop of the algorithm.

5.6.1 Main loop of the algorithm

The main loop of the parsing algorithm proceeds as follows. The next token of the program is read. Depending on the token value there are three possibilities:

- Token is equal to *system* – name of the module is read, a new module of the given name is created in the tool's model structure, and the module is set as the root element of the system. At this point parsing of the datapath body is started.

- Token is equal *dp* - datapath declaration processing starts. Name of the module is read and an advance to the next token is made. There are two possibilities at this point. If the token is equal to ":", it means that coping datapath procedure was used in the GEZEL program. In this case name of the datapath supposed to be cloned is read and new instance for corresponding type of module is added to the model. Otherwise declaration of completely new datapath is expected. A new module is added to the model representation in the tool structure. If the token is equal to "(", first list of parameters has to be parsed. Using obtained name, parameter type (input or output) and value type for each element from the list, corresponding objects are added to the datapath of the newly created module. Next step is to parse the body of the datapath.

- Token is equal to *fsm* – declaration of controller is expected. First, application reads name of the datapath to which given controller is attached. New controller in the tool hardware model representation is created and added to the appropriate module - the same module in which datapath that corresponds to the attached GEZEL datapath exists. Next step is to parse the body of the controller declaration.

5.6.2 Parsing of the datapath

Inside datapath body, parsing algorithm proceeds as follows:

- If token equals to *reg* or *sig* – declaration of signal or register is processed. Name and value type are parsed. Finally, new element is created in the tool representation of the datapath.

- Token is equal to *lookup* – lookup table declaration. Name, type and values are parsed. Corresponding element is created in the tool representation of datapath.

- Token equals to *sfg* – signal flow graph declaration is expected. After name is parsed, new corresponding signal flow graph is created and added to the tool representation of the datapath. Next, each expression is parsed one by one. Left side element is read and expression parsing algorithm is applied on the right side of the equation. After successful parsing operation, new expression is created and added to the appropriate signal flow graph in the tool representation of the hardware model.

- Token equals to *always* – similar to parsing *sfg*. First difference is that there is no name to parse, so created signal flow graph in the tool model is named always. Second difference is that after the whole program is parsed, signal flow graph *always* is added to the list of launched signal flow graphs for every transition belonging to the given module.

- Token equals to *use* – indicates beginning of sub-module use declaration. Name of the sub-module is read together with parameter list. Structure describing use of sub-module is created and added inside the tool representation of the datapath (instance of UseModule class).

5.6.3 Parsing of the controller

The GEZEL parsing algorithm inside declaration of the controller body proceeds as follows. The next token is read and on the basis of its value there are several cases:

- Token equals *initial* – declaration of initial state. Name of the state is parsed and new state is added to the controller inside the tool structure. The state is set as an initial state for the controller.

- Token equals *state* – declaration of states. Names are parsed. Corresponding states are created and added to the controller.

- Token equals '@' character – indicates transition processing. Name of the state is parsed. Next, if transition depends on conditions – each option is treated as separate case. Each condition is parsed using expression parsing algorithm, the same which is used in parsing of signal flow graph expressions. Otherwise, transition is stored as a single case with condition set to true. Next, signal flow graphs launched during the transition and the next state declaration are parsed. Finally, transition is created and added to the appropriate state in the tool representation of the model.

5.6.4 Parsing of GEZEL expressions

Expressions are stored in a form of tree structure, in which operators are branches and variables and constants are leafs. In order to create such a structure recursive descent parsing is used. In each branch, type of the operator (addition, multiplication, etc.) is defined and links to its operands.

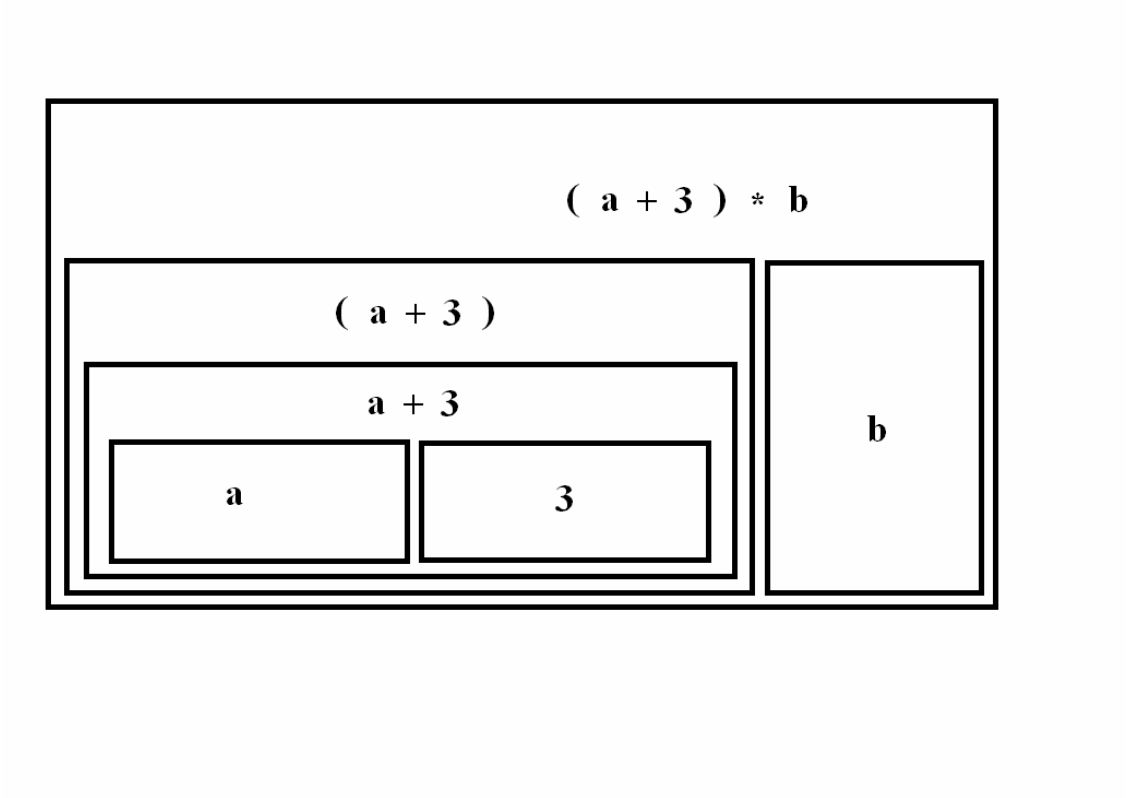

Figure 5.13: An example of the expression structure.

Expression parsing algorithm:

- Find operator in the expression that binds the least and is not enclosed in any brackets. When such an operator is found, program divides the whole expression into parts. Number of parts is equal to the arity of the operator. For each operand (part of the expression) the same expression parsing algorithm is called recurrently. Results of the applying algorithm on operands are bind together with type of operator.

- If there are no operators, then it means expression must be enclosed in brackets, be a single constant or variable. If expression is in brackets, unary operator of appropriate type is created and algorithm is recurrently called on the expression inside the brackets. Otherwise, in case of variable or constant, a structure that represents appropriate element is created and algorithm returns to the creation of the parent operator.

6 Conclusions

In this thesis a graphical tool for modeling and analysis of hardware architectures was developed and presented. The tool allows visualization of models created in GEZEL-like languages, assists in development and provides means for verification.

All initial project objectives as specified in the chapter 1 (see page 9) were met.

- A graphical representation for high level hardware description languages like GEZEL was designed. Each building element used in the designed representation models exactly one construct from GEZEL, which makes it easy to understand for someone already familiar with this language.

- A tool that allows creation and edition of hardware models in a user-friendly graphical environment was developed. Some additional functionality that exceeds initial projects objectives was integrated in the tool, namely check for well-formedness of the model.

- Finally, possibilities in the area of formal verification of models developed using the tool were investigated. A component capable of creation NuSMV models was created, allowing usage of this symbolic model checker as a back-end for full functional check.

The tool developed as a part of the thesis is able to import hardware models from GEZEL files. Except IP blocks, the whole language is covered by the parsing mechanism. If a particular construct from the GEZEL language does not have a corresponding element in the tool internal representation, the construct is modeled using available elements. For example this is a case for hardwired controller and sequencer, which are represented by an appropriate composition of controller states and transitions. Except GEZEL sources, the tool is able to use two kinds of dedicated XML files for serialization, one for hardware models alone and one for models together with its graphical representation data.

The well-formedness check routine is included in the tool. It is performed on the model after each significant modification, and in case of errors the user is instantly notified about their presence. This verification strategy can be really handy for the developer, by assisting in spotting errors otherwise hard to notice. It has proven its usefulness during

testing of the tool when various hardware descriptions written in GEZEL language were imported. The tool was able to point out some errors present in the implementation of the Euclid algorithm available on the GEZEL home page. The well-formedness check mechanism lacks detection of combinatorial loops in the model, which could be resolved using the graph theory. However, the project's limited time frame did not allow implementing this functionality.

Full functional check provided by the tool through the use of NusSMV as a back-end was successfully performed on small models such as implementation of Euclid algorithm. It was not possible to verify correctness of Simplified DES. Surprisingly it did not outperform verification using UPPAAL presented in [2] and results were approximately the same. However, the results of verification in terms of NuSMV performance as a back-end for model checking are inconclusive, and this area needs more investigation and experiments. It is possible that different approach in translation of hardware descriptions to NuSMV will lead to more optimalized models and much better performance of verification.

Neither full functional nor structural test of the tool was performed. The reason for that was lack of time for such an activity. During development and debugging different components of the tool were partially tested. Several hardware descriptions available on the GEZEL home page were successfully imported and some models were created from scratch by hand. However, presence of some minor errors cannot be ruled out.

It was observed that some parts of the tool could be done in a better, more standardized way. In particular such a part is the component responsible for reading GEZEL programs into the tool structure. Instead of providing ad-hoc but still sound solution, tools dedicated for this purpose, like Lex and Yacc, could be used.

Summarizing, the application developed as a part of the thesis, if the part that visualizes and allows edition of hardware descriptions is concerned, is close to a commercial product. It can assist in creation of new and modification of already developed GEZEL descriptions. Obtained models can be translated to GEZEL, from which hardware can be synthesized. However, the component responsible for full functional check needs further investigation and development in order to become a useful tool.

References

[1] Patrick Schaumont, Doris Ching, Herwin Chan, Jørgen Steensgaard-Madsen, Andreas Vad-Lorentzen, Eric Simpson. GEZEL User Manual.
http://rijndael.ece.vt.edu/gezel2/index.php/GEZEL_User_Manual

[2] Michael R. Hansen, Jan Madsen, Aske Wiid Brekling. Semantics and Verification of a Language for Modeling Hardware Architectures. In Formal Methods and Hybrid Real-Time Systems, Lecture Notes in Computer Science, vol. 4700, Springer 2007, pp 300 – 319.

[3] Sun Microsystems. Java 2 Platform, Standard Edition, v 1.4.2. API Specification.
http://java.sun.com/j2se/1.4.2/docs/api/

[4] JGraph Ltd. JGraph and JGraph Layout Pro User Manual. David Benson 2004 – 2007.
http://www.jgraph.com/pub/jgraphmanual.pdf

[5] Roberto Cavada, Alessandro Cimatti, Charles Arthur Jochim, Gavin Keighren, Emanuele Olivetti, Marco Pistore, Marco Roveri and Andrei Tchaltsev. NuSMV 2.4 User Manual. 1998 – 2005 by CMU and ITC-irst.
http://nusmv.fbk.eu/NuSMV/userman/v24/nusmv.pdf

[6] Roberto Cavada, Alessandro Cimatti, Gavin Keighren, Emanuele Olivetti, Marco Pistore and Marco Roveri. NuSMV 2.2 Tutorial.
http://nusmv.fbk.eu/NuSMV/tutorial/v24/tutorial.pdf

[7] Intel. Moore's Law page.
http://www.intel.com/technology/mooreslaw/index.htm

Appendix A

A.1 GEZEL sources

A.1.1 Euclid algorithm

```
dp euclid(in  m_in, n_in : ns(16);
          out gcd        : ns(16)) {
  reg m, n               : ns(16);
  reg done               : ns(1);
  reg factor             : ns(16);

  sfg init    { m = m_in; n = n_in; factor = 0;
                $display("cycle=", $cycle, " m=", m_in, " n=", n_in); }
  sfg shiftm  { m = m >> 1; }
  sfg shiftn  { n = n >> 1; }
  sfg reduce  { m = (m >= n) ? m - n : m;
                n = (n >  m) ? n - m : n; }
  sfg shiftf  { factor = factor + 1; }
  sfg outidle { gcd = 0; done = ((m == 0) | (n == 0)); }
  sfg complete{ gcd = ((m > n) ? m : n) << factor;
                $display("cycle=", $cycle, " gcd=", gcd); }
}

fsm euclid_ctl(euclid) {
  initial s0;
  state s1, s2;

  @s0 (init, outidle) -> s1;
  @s1 if (done)                then (complete)          -> s2;
      else if ( m[0] &  n[0]) then (reduce, outidle)   -> s1;
      else if ( m[0] & ~n[0]) then (shiftn, outidle)   -> s1;
      else if (~m[0] &  n[0]) then (shiftm, outidle)   -> s1;
                              else (shiftn, shiftm,
                                    shiftf, outidle)   -> s1;
  @s2 (outidle) -> s2;
}

//--- Testbench
dp test_euclid(out m, n : ns(16)) {
  always {
    m = 2322;
    n = 654;
  }
}
```

```
dp euclid_sys {
   sig m, n, gcd : ns(16);
   use euclid(m, n, gcd);
   use test_euclid(m, n);
}

system S {
  euclid_sys;
}
```

A.1.2 Simplified DES

```
dp encrypt(in in_port         : ns(8);
      in in_key          : ns(10);
      out out_port       : ns(8);
      out done           : ns(1)){

      reg input, k1, k2, res, fk, sw          : ns(8);
      reg key                                 : ns(10);
      reg left, right                         : ns(4);

      sig ls1, ls2, pk                        : ns(10);
      sig p4, fkl                             : ns(4);
      sig ep                                  : ns(8);
      sig s0r, s0c, s1r, s1c, first, second   : ns(2);

      lookup s00 : ns(2) = {1, 0, 3, 2};
      lookup s01 : ns(2) = {3, 1, 2, 0};
      lookup s02 : ns(2) = {0, 2, 1, 3};
      lookup s03 : ns(2) = {3, 1, 3, 2};
      lookup s10 : ns(2) = {0, 1, 2, 3};
      lookup s11 : ns(2) = {2, 0, 1, 3};
      lookup s12 : ns(2) = {3, 0, 1, 0};
      lookup s13 : ns(2) = {2, 1, 0, 3};

      sfg ini {
            input = in_port;
            key = in_key;
            out_port = 0;
            done = 0;
            $display("cycle:", $cycle, $dec, ", in:", in_port);
      }

      sfg keys {
            pk = key[5]#key[7]#key[8]#key[0]#key[9]#key[3]#key[6]#key[1]
                  #key[4]#key[2];
            ls1 = pk[8]#pk[7]#pk[6]#pk[5]#pk[9]#pk[3]#pk[2]#pk[1]
                  #pk[0]#pk[4];
            ls2 = s1[7]#ls1[6]#ls1[5]#ls1[9]#ls1[8]#ls1[2]#ls1[1]#ls1[0]
                  #ls1[4]#ls1[3];
            k1 = s1[4]#ls1[7]#ls1[3]#ls1[6]#ls1[2]#ls1[5]#ls1[0]#ls1[1];
            k2 = ls2[4]#ls2[7]#ls2[3]#ls2[6]#ls2[2]#ls2[5]
                  #ls2[0]#ls2[1];
            out_port = 0;
```

```
      done = 0;
}

sfg lr {
      left=input[6]#input[2]#input[5]#input[7];
      right=input[4]#input[0]#input[3]#input[1];
      out_port = 0;
      done = 0;
}

sfg ip {
      ep = right[0]#right[3]#right[2]#right[1]#right[2]#right[1]
            #right[0]#right[3];
      res = ep ^ k1;
      out_port = 0;
      done = 0;
}

sfg slook {
      s0r = res[7]#res[4];
      s0c = res[6]#res[5];
      s1r = res[0]#res[3];
      s1c = res[2]#res[1];

      first=(s0r==0)?s00(s0c):(s0r==1)?s01(s0c):(s0r==2)?s02(s0c):
            s03(s0c);

      second=(s1r==0)?s10(s1c):(s1r==1)?s11(s1c):(s1r==2)?s12(s1c)
      :
            s13(s1c);

      p4 = first[0]#second[0]#second[1]#first[1];
      fkl = left ^ p4;
      fk = fkl#right;
      sw = right#fkl;
      out_port = 0;
      done = 0;
}

sfg rl {
      left = sw[7]#sw[6]#sw[5]#sw[4];
      right = sw[3]#sw[2]#sw[1]#sw[0];
      out_port = 0;
      done = 0;
}

sfg secondf {
      ep = right[0]#right[3]#right[2]#right[1]#right[2]#right[1]
            #right[0]#right[3];
      res = ep ^ k2;
      out_port = 0;
      done = 0;
}

sfg setout {
      out_port = fk[4]#fk[7]#fk[5]#fk[3]#fk[1]#fk[6]#fk[0]#fk[2];
      done = 1;
```

```
      }
}

fsm encrypt_ctl(encrypt) {
      initial s0;
      state s1, s2, s3, s4, s5, s6, s7, s8, s9;

      @s0 (ini) -> s1;
      @s1 (keys) -> s2;
      @s2 (lr) -> s3;
      @s3 (ip) -> s4;
      @s4 (slook) -> s5;
      @s5 (rl) -> s6;
      @s6 (secondf) -> s7;
      @s7 (slook) -> s8;
      @s8 (setout) -> s9;
      @s9 (setout) -> s9;
}

dp decrypt(in in_port         : ns(8);
           in in_key          : ns(10);
           in in_start        : ns(1);
           out out_port       : ns(8);
           out done           : ns(1)){

      reg input, k1, k2, res, fk, sw     : ns(8);
      reg key                            : ns(10);
      reg left, right                    : ns(4);
      reg start                          : ns(1);

      sig ls1, ls2, pk                        : ns(10);
      sig p4, fkl                             : ns(4);
      sig ep                                  : ns(8);
      sig s0r, s0c, s1r, s1c, first, second   : ns(2);

      lookup s00 : ns(2) = {1, 0, 3, 2};
      lookup s01 : ns(2) = {3, 1, 2, 0};
      lookup s02 : ns(2) = {0, 2, 1, 3};
      lookup s03 : ns(2) = {3, 1, 3, 2};
      lookup s10 : ns(2) = {0, 1, 2, 3};
      lookup s11 : ns(2) = {2, 0, 1, 3};
      lookup s12 : ns(2) = {3, 0, 1, 0};
      lookup s13 : ns(2) = {2, 1, 0, 3};

      sfg pre { start = in_start;
            out_port = 0;
            done = 0;
      }

      sfg ini {
            input = in_port;
            key = in_key;
            out_port = 0;
            done = 0;
      }

      sfg keys {
```

```
      pk = key[5]#key[7]#key[8]#key[0]#key[9]#key[3]#key[6]#key[1]
            #key[4]#key[2];
      ls1 = pk[8]#pk[7]#pk[6]#pk[5]#pk[9]#pk[3]#pk[2]#pk[1]
            #pk[0]#pk[4];
      ls2 = ls1[7]#ls1[6]#ls1[5]#ls1[9]#ls1[8]#ls1[2]#ls1[1]
            #ls1[0]#ls1[4]#ls1[3];
      k1 = ls1[4]#ls1[7]#ls1[3]#ls1[6]#ls1[2]#ls1[5]
            #ls1[0]#ls1[1];
      k2 = ls2[4]#ls2[7]#ls2[3]#ls2[6]#ls2[2]#ls2[5]
            #ls2[0]#ls2[1];
      out_port = 0;
      done = 0;
}

sfg lr {
      left=input[6]#input[2]#input[5]#input[7];
      right=input[4]#input[0]#input[3]#input[1];
      out_port = 0;
      done = 0;
}

sfg ip {
      ep = right[0]#right[3]#right[2]#right[1]#right[2]#right[1]
            #right[0]#right[3];
      res = ep ^ k2;
      out_port = 0;
      done = 0;
}

sfg slook {
      s0r = res[7]#res[4];
      s0c = res[6]#res[5];
      s1r = res[0]#res[3];
      s1c = res[2]#res[1];

      first=(s0r==0)?s00(s0c):(s0r==1)?s01(s0c):(s0r==2)?s02(s0c):
            s03(s0c);

      second=(s1r==0)?s10(s1c):(s1r==1)?s11(s1c):(s1r==2)?s12(s1c)
      :
            s13(s1c);

      p4 = first[0]#second[0]#second[1]#first[1];
      fkl = left ^ p4;
      fk = fkl#right;
      sw = right#fkl;
      out_port = 0;
      done = 0;
}

sfg rl {
      left = sw[7]#sw[6]#sw[5]#sw[4];
      right = sw[3]#sw[2]#sw[1]#sw[0];
      out_port = 0;
      done = 0;
}
```

```
      sfg secondf {
            ep = right[0]#right[3]#right[2]#right[1]#right[2]#right[1]
                  #right[0]#right[3];
            res = ep ^ k1;
            out_port = 0;
            done = 0;
      }

      sfg setout  {
            out_port = fk[4]#fk[7]#fk[5]#fk[3]#fk[1]#fk[6]#fk[0]#fk[2];
            done = 1;
            $display("cycle:", $cycle, $dec, ", out:", out_port);
      }
}

fsm decrypt_ctl(decrypt) {
      initial s0;
      state s1, s2, s3, s4, s5, s6, s7, s8, s9;

      @s0 if (start)    then  (ini) -> s1;
            else  (pre) -> s0;
      @s1 (keys) -> s2;
      @s2 (lr) -> s3;
      @s3 (ip) -> s4;
      @s4 (slook) -> s5;
      @s5 (rl) -> s6;
      @s6 (secondf) -> s7;
      @s7 (slook) -> s8;
      @s8 (setout) -> s9;
      @s9 (setout) -> s9;
}

dp test_encrypt(out n : ns(8); out key : ns(10)){
      always {
            n=100;
            key=2;
      }
}


dp test_sys {
      sig n, o, oo      : ns(8);
      sig k       : ns(10);
      sig d, dd   : ns(1);
      sig s       : ns(1);

      use test_encrypt(n,k);
      use encrypt(n,k,o,d);
      use decrypt(o,k,d,oo,dd);
}

system S {
      test_sys;
}
```

A.2 XML structure

A.2.1 Euclid algorithm – without project settings

```
<model>
  <modules>
    <module>
      <name>euclid_sys</name>
      <datapath>
        <lookupTables/>
        <elements>
          <element>
            <name>m</name>
            <elementType>signal</elementType>
            <valueType>ns(16)</valueType>
          </element>
          <element>
            <name>n</name>
            <elementType>signal</elementType>
            <valueType>ns(16)</valueType>
          </element>
          <element>
            <name>gcd</name>
            <elementType>signal</elementType>
            <valueType>ns(16)</valueType>
          </element>
        </elements>
        <sfgs/>
        <useModules>
          <useModule>
            <module>
              <name>euclid</name>
              <datapath>
                <lookupTables/>
                <elements>
                  <element>
                    <name>m_in</name>
                    <elementType>input</elementType>
                    <valueType>ns(16)</valueType>
                  </element>
                  <element>
                    <name>n_in</name>
                    <elementType>input</elementType>
                    <valueType>ns(16)</valueType>
                  </element>
                  <element>
                    <name>gcd</name>
                    <elementType>output</elementType>
                    <valueType>ns(16)</valueType>
                  </element>
                  <element>
                    <name>m</name>
                    <elementType>register</elementType>
                    <valueType>ns(16)</valueType>
                  </element>
                  <element>
```

```
    <name>n</name>
    <elementType>register</elementType>
    <valueType>ns(16)</valueType>
  </element>
  <element>
    <name>done</name>
    <elementType>register</elementType>
    <valueType>ns(1)</valueType>
  </element>
  <element>
    <name>factor</name>
    <elementType>register</elementType>
    <valueType>ns(16)</valueType>
  </element>
</elements>
<sfgs>
  <sfg>
    <name>init</name>
    <expressions>
      <expression>
        <leftSideElement reference=
           "../../../../../elements/element[4]"/>
        <rightSideElements>
          <element reference=
              "../../../../../../elements/element"/>
        </rightSideElements>
        <id>0</id>
        <expression class="variable">
          <element reference=
              "../../../../../../elements/element"/>
          <type>20</type>
        </expression>
      </expression>
      <expression>
        <leftSideElement reference=
              "../../../../../elements/element[5]"/>
        <rightSideElements>
          <element reference=
              "../../../../../../elements/element[2]"/>
        </rightSideElements>
        <id>0</id>
        <expression class="variable">
          <element reference=
              "../../../../../../elements/element[2]"/>
          <type>20</type>
        </expression>
      </expression>
      <expression>
        <leftSideElement reference=
           "../../../../../elements/element[7]"/>
        <rightSideElements/>
        <id>0</id>
        <expression class="constant">
          <value>0</value>
          <type>19</type>
        </expression>
      </expression>
```

```
  </expressions>
</sfg>
<sfg>
  <name>shiftm</name>
  <expressions>
    <expression>
      <leftSideElement reference=
         "../../../../../elements/element[4]"/>
      <rightSideElements>
        <element reference=
           "../../../../../../elements/element[4]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="binaryOperator">
        <expression1 class="variable">
          <element reference="../../../../../
            ../../elements/element[4]"/>
          <type>20</type>
        </expression1>
        <expression2 class="constant">
          <value>1</value>
          <type>19</type>
        </expression2>
        <type>16</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>shiftn</name>
  <expressions>
    <expression>
      <leftSideElement reference=
         "../../../../../elements/element[5]"/>
      <rightSideElements>
        <element reference=
           "../../../../../../elements/element[5]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="binaryOperator">
        <expression1 class="variable">
          <element reference="../../../
            ../../../../elements/element[5]"/>
          <type>20</type>
        </expression1>
        <expression2 class="constant">
          <value>1</value>
          <type>19</type>
        </expression2>
        <type>16</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>reduce</name>
  <expressions>
```

```
<expression>
  <leftSideElement reference=
     "../../../../../elements/element[4]"/>
  <rightSideElements>
    <element reference=
       "../../../../../../elements/element[4]"/>
    <element reference=
       "../../../../../../elements/element[5]"/>
  </rightSideElements>
  <id>0</id>
  <expression class="ternaryOperator">
    <expression1 class="UnaryOperator">
      <expression class="binaryOperator">
        <expression1 class="variable">
          <element reference="../../../../../../
              ../../../elements/element[4]"/>
          <type>20</type>
        </expression1>
        <expression2 class="variable">
          <element reference="../../../../../../
              ../../../elements/element[5]"/>
          <type>20</type>
        </expression2>
        <type>23</type>
      </expression>
      <type>10</type>
    </expression1>
    <expression2 class="binaryOperator">
      <expression1 class="variable">
        <element reference="../../../../../
              ../../../elements/element[4]"/>
        <type>20</type>
      </expression1>
      <expression2 class="variable">
        <element reference="../../../../../
              ../../../elements/element[5]"/>
        <type>20</type>
      </expression2>
      <type>18</type>
    </expression2>
    <expression3 class="variable">
      <element reference="../../../../
        ../../../elements/element[4]"/>
      <type>20</type>
    </expression3>
    <type>14</type>
  </expression>
</expression>
<expression>
  <leftSideElement reference=
     "../../../../../elements/element[5]"/>
  <rightSideElements>
    <element reference=
       "../../../../../../elements/element[5]"/>
    <element reference=
       "../../../../../../elements/element[4]"/>
  </rightSideElements>
```

```
      <id>0</id>
      <expression class="ternaryOperator">
        <expression1 class="UnaryOperator">
          <expression class="binaryOperator">
            <expression1 class="variable">
              <element reference="../../../../../
                  ../../../../elements/element[5]"/>
              <type>20</type>
            </expression1>
            <expression2 class="variable">
              <element reference="../../../../../../
                  ../../../elements/element[4]"/>
              <type>20</type>
            </expression2>
            <type>25</type>
          </expression>
          <type>10</type>
        </expression1>
        <expression2 class="binaryOperator">
          <expression1 class="variable">
            <element reference="../../../../../
                 ../../../elements/element[5]"/>
            <type>20</type>
          </expression1>
          <expression2 class="variable">
            <element reference="../../../../../
                 ../../../elements/element[4]"/>
            <type>20</type>
          </expression2>
          <type>18</type>
        </expression2>
        <expression3 class="variable">
          <element reference="../../../../../
                 ../../elements/element[5]"/>
          <type>20</type>
        </expression3>
        <type>14</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>shiftf</name>
  <expressions>
    <expression>
      <leftSideElement reference="../../../../
           ../elements/element[7]"/>
      <rightSideElements>
        <element reference="../../../../../
            ../elements/element[7]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="binaryOperator">
        <expression1 class="variable">
          <element reference="../../../../../../
            ../elements/element[7]"/>
          <type>20</type>
```

```
        </expression1>
        <expression2 class="constant">
          <value>1</value>
          <type>19</type>
        </expression2>
        <type>17</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>outidle</name>
  <expressions>
    <expression>
      <leftSideElement reference="../../../../../
           elements/element[3]"/>
      <rightSideElements/>
      <id>0</id>
      <expression class="constant">
        <value>0</value>
        <type>19</type>
      </expression>
    </expression>
    <expression>
      <leftSideElement reference="../../../../../
           elements/element[6]"/>
      <rightSideElements>
        <element reference="../../../../../
            ../elements/element[4]"/>
        <element reference="../../../../../
            ../elements/element[5]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="UnaryOperator">
        <expression class="binaryOperator">
          <expression1 class="UnaryOperator">
            <expression class="binaryOperator">
              <expression1 class="variable">
                <element reference="../../../../../
                 ../../../../../../elements/element[4]"/>
                <type>20</type>
              </expression1>
              <expression2 class="constant">
                <value>0</value>
                <type>19</type>
              </expression2>
              <type>21</type>
            </expression>
            <type>10</type>
          </expression1>
          <expression2 class="UnaryOperator">
            <expression class="binaryOperator">
              <expression1 class="variable">
                <element reference="../../../../../
                 ../../../../../../elements/element[5]"/>
                <type>20</type>
              </expression1>
```

```
              <expression2 class="constant">
                <value>0</value>
                <type>19</type>
              </expression2>
              <type>21</type>
            </expression>
            <type>10</type>
          </expression2>
          <type>13</type>
        </expression>
        <type>10</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>complete</name>
  <expressions>
    <expression>
      <leftSideElement reference="../../../../
           ../elements/element[3]"/>
      <rightSideElements>
        <element reference="../../../../../
            ../elements/element[4]"/>
        <element reference="../../../../../
            ../elements/element[5]"/>
        <element reference="../../../../../
            ../elements/element[7]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="binaryOperator">
        <expression1 class="UnaryOperator">
          <expression class="ternaryOperator">
            <expression1 class="UnaryOperator">
              <expression class="binaryOperator">
                <expression1 class="variable">
                  <element reference="../../../../../../
                       ../../../../../../
                       elements/element[4]"/>
                  <type>20</type>
                </expression1>
                <expression2 class="variable">
                  <element reference="../../../../../../
                       ../../../../../../
                       elements/element[5]"/>
                  <type>20</type>
                </expression2>
                <type>25</type>
              </expression>
              <type>10</type>
            </expression1>
            <expression2 class="variable">
              <element reference="../../../../../
                  ../../../../elements/element[4]"/>
              <type>20</type>
            </expression2>
            <expression3 class="variable">
```

```
                    <element reference="../../../../../
                        ../../../../elements/element[5]"/>
                    <type>20</type>
                  </expression3>
                  <type>14</type>
                </expression>
                <type>10</type>
              </expression1>
              <expression2 class="variable">
                <element reference="../../../../../
                  ../../elements/element[7]"/>
                <type>20</type>
              </expression2>
              <type>15</type>
            </expression>
          </expression>
        </expressions>
      </sfg>
    </sfgs>
    <useModules/>
    <parentModule reference="../.."/>
  </datapath>
  <fsm>
    <initialState>
      <name>s0</name>
      <transitions>
        <fsmTransition>
          <condition>
            <expression class="constant">
              <value>1</value>
              <type>19</type>
            </expression>
          </condition>
          <usedSfgs>
            <sfg reference="../../../../../
                 ../datapath/sfgs/sfg"/>
            <sfg reference="../../../../../
                 ../datapath/sfgs/sfg[6]"/>
          </usedSfgs>
          <nextState>
            <name>s1</name>
            <transitions>
              <fsmTransition>
                <condition>
                  <expression class="variable">
                    <element reference="../../../../../
                        ../../../../../
                        datapath/elements/element[6]"/>
                    <type>20</type>
                  </expression>
                </condition>
                <usedSfgs>
                  <sfg reference="../../../../../
                       ../../../../datapath/sfgs/sfg[7]"/>
                </usedSfgs>
                <nextState>
                  <name>s2</name>
```

```
    <transitions>
      <fsmTransition>
        <condition>
          <expression class="constant">
            <value>1</value>
            <type>19</type>
          </expression>
        </condition>
        <usedSfgs>
          <sfg reference="../../../../../
                ../../../../../../../
               datapath/sfgs/sfg[6]"/>
        </usedSfgs>
        <nextState reference="../../.."/>
      </fsmTransition>
    </transitions>
  </nextState>
</fsmTransition>
<fsmTransition>
  <condition>
    <expression class="binaryOperator">
      <expression1 class="bitSelection">
        <element reference="../../../../../
          ../../../../../../
          datapath/elements/element[4]"/>
        <firstBit>0</firstBit>
        <lastBit>0</lastBit>
        <type>26</type>
      </expression1>
      <expression2 class="bitSelection">
        <element reference="../../../../../
          ../../../../../../
          datapath/elements/element[5]"/>
        <firstBit>0</firstBit>
        <lastBit>0</lastBit>
        <type>26</type>
      </expression2>
      <type>12</type>
    </expression>
  </condition>
  <usedSfgs>
    <sfg reference="../../../../../
         ../../../../
         datapath/sfgs/sfg[4]"/>
    <sfg reference="../../../../../
         ../../../../
         datapath/sfgs/sfg[6]"/>
  </usedSfgs>
  <nextState reference="../../.."/>
</fsmTransition>
<fsmTransition>
  <condition>
    <expression class="binaryOperator">
      <expression1 class="bitSelection">
        <element reference="../../../../../
          ../../../../../../
          datapath/elements/element[4]"/>
```

```
          <firstBit>0</firstBit>
          <lastBit>0</lastBit>
          <type>26</type>
        </expression1>
        <expression2 class="UnaryOperator">
          <expression class="bitSelection">
            <element reference="../../../../../
                  ../../../../../../../datapath/
                  elements/element[5]"/>
            <firstBit>0</firstBit>
            <lastBit>0</lastBit>
            <type>26</type>
          </expression>
          <type>27</type>
        </expression2>
        <type>12</type>
      </expression>
    </condition>
    <usedSfgs>
      <sfg reference="../../../../../
           ../../../../datapath/
           sfgs/sfg[3]"/>
      <sfg reference="../../../../../
           ../../../../datapath/
           sfgs/sfg[6]"/>
    </usedSfgs>
    <nextState reference="../../.."/>
  </fsmTransition>
  <fsmTransition>
    <condition>
      <expression class="binaryOperator">
        <expression1 class="UnaryOperator">
          <expression class="bitSelection">
            <element reference="../../../../../
                  ../../../../../../../datapath/
                  elements/element[4]"/>
            <firstBit>0</firstBit>
            <lastBit>0</lastBit>
            <type>26</type>
          </expression>
          <type>27</type>
        </expression1>
        <expression2 class="bitSelection">
          <element reference="../../../../../
            ../../../../../../
            datapath/elements/element[5]"/>
          <firstBit>0</firstBit>
          <lastBit>0</lastBit>
          <type>26</type>
        </expression2>
        <type>12</type>
      </expression>
    </condition>
    <usedSfgs>
      <sfg reference="../../../../../
           ../../../../datapath/
           sfgs/sfg[2]"/>
```

```
                    <sfg reference="../../../../../
                         ../../../../datapath/
                         sfgs/sfg[6]"/>
                  </usedSfgs>
                  <nextState reference="../../.."/>
                </fsmTransition>
                <fsmTransition>
                  <condition>
                    <expression class="constant">
                      <value>1</value>
                      <type>19</type>
                    </expression>
                  </condition>
                  <usedSfgs>
                    <sfg reference="../../../../../
                         ../../../../datapath/
                        sfgs/sfg[3]"/>
                    <sfg reference="../../../../../
                         ../../../../datapath/
                         sfgs/sfg[2]"/>
                    <sfg reference="../../../../../
                         ../../../../datapath/
                         sfgs/sfg[5]"/>
                    <sfg reference="../../../../../
                         ../../../../datapath/
                         sfgs/sfg[6]"/>
                  </usedSfgs>
                  <nextState reference="../../.."/>
                </fsmTransition>
              </transitions>
            </nextState>
          </fsmTransition>
        </transitions>
      </initialState>
      <states>
        <fsmState reference="../../initialState"/>
        <fsmState reference="../../initialState/transitions/
             fsmTransition/nextState"/>
        <fsmState reference="../../initialState/transitions/
             fsmTransition/nextState/transitions/
             fsmTransition/nextState"/>
      </states>
    </fsm>
    <parentModel reference="../../../../../../.."/>
  </module>
  <parameters>
    <element reference="../../../../elements/element"/>
    <element reference="../../../../elements/element[2]"/>
    <element reference="../../../../elements/element[3]"/>
  </parameters>
  <name>euclid</name>
</useModule>
<useModule>
  <module>
    <name>test_euclid</name>
    <datapath>
      <lookupTables/>
```

```
  <elements>
    <element>
      <name>m</name>
      <elementType>output</elementType>
      <valueType>ns(16)</valueType>
    </element>
    <element>
      <name>n</name>
      <elementType>output</elementType>
      <valueType>ns(16)</valueType>
    </element>
  </elements>
  <sfgs>
    <sfg>
      <name>always</name>
      <expressions>
        <expression>
          <leftSideElement reference="../../../../../
               elements/element"/>
          <rightSideElements/>
          <id>0</id>
          <expression class="constant">
            <value>2322</value>
            <type>19</type>
          </expression>
        </expression>
        <expression>
          <leftSideElement reference="../../../../../
               elements/element[2]"/>
          <rightSideElements/>
          <id>0</id>
          <expression class="constant">
            <value>654</value>
            <type>19</type>
          </expression>
        </expression>
      </expressions>
    </sfg>
  </sfgs>
  <useModules/>
  <parentModule reference="../.."/>
</datapath>
<fsm>
  <states>
    <fsmState>
      <name>s0</name>
      <transitions>
        <fsmTransition>
          <condition>
            <expression class="constant">
              <value>1</value>
              <type>19</type>
            </expression>
          </condition>
          <usedSfgs>
            <sfg reference="../../../../../../../
                datapath/sfgs/sfg"/>
```

```
                        </usedSfgs>
                        <nextState reference="../../.."/>
                      </fsmTransition>
                    </transitions>
                  </fsmState>
                </states>
              </fsm>
              <parentModel reference="../../../../../../.."/>
            </module>
            <parameters>
              <element reference="../../../../elements/element"/>
              <element reference="../../../../elements/element[2]"/>
            </parameters>
            <name>test_euclid</name>
          </useModule>
        </useModules>
        <parentModule reference="../.."/>
      </datapath>
      <fsm>
        <states/>
      </fsm>
      <parentModel reference="../../.."/>
    </module>
    <module reference="../module/datapath/useModules/
      useModule/module"/>
    <module reference="../module/datapath/useModules/
      useModule[2]/module"/>
  </modules>
  <systemModule reference="../modules/module"/>
  <ExpressionCounter>0</ExpressionCounter>
</model>
```

A.2.2 Euclid algorithm – with project settings

```
<maaoha.view.serialization.ModelData>
  <model>
    <modules>
      <module>
        <name>euclid_sys</name>
        <datapath>
          <lookupTables/>
          <elements>
            <element>
              <name>m</name>
              <elementType>signal</elementType>
              <valueType>ns(5)</valueType>
            </element>
            <element>
              <name>n</name>
              <elementType>signal</elementType>
              <valueType>ns(5)</valueType>
            </element>
            <element>
              <name>gcd</name>
              <elementType>signal</elementType>
```

```
    <valueType>ns(5)</valueType>
  </element>
</elements>
<sfgs/>
<useModules>
  <useModule>
    <module>
      <name>euclid</name>
      <datapath>
        <lookupTables/>
        <elements>
          <element>
            <name>m_in</name>
            <elementType>input</elementType>
            <valueType>ns(5)</valueType>
          </element>
          <element>
            <name>n_in</name>
            <elementType>input</elementType>
            <valueType>ns(5)</valueType>
          </element>
          <element>
            <name>gcd</name>
            <elementType>output</elementType>
            <valueType>ns(5)</valueType>
          </element>
          <element>
            <name>m</name>
            <elementType>register</elementType>
            <valueType>ns(5)</valueType>
          </element>
          <element>
            <name>n</name>
            <elementType>register</elementType>
            <valueType>ns(5)</valueType>
          </element>
          <element>
            <name>done</name>
            <elementType>register</elementType>
            <valueType>ns(1)</valueType>
          </element>
          <element>
            <name>factor</name>
            <elementType>register</elementType>
            <valueType>ns(5)</valueType>
          </element>
        </elements>
        <sfgs>
          <sfg>
            <name>init</name>
            <expressions>
              <expression>
                <leftSideElement reference="../../../../../
              elements/element[4]"/>
                <rightSideElements>
                  <element reference="../../../../../../
              elements/element"/>
```

```
      </rightSideElements>
      <id>0</id>
      <expression class="variable">
        <element reference="../../../../../../
    elements/element"/>
        <type>20</type>
      </expression>
    </expression>
    <expression>
      <leftSideElement reference="../../../../../
    elements/element[5]"/>
      <rightSideElements>
        <element reference="../../../../../../
    elements/element[2]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="variable">
        <element reference="../../../../../../
    elements/element[2]"/>
        <type>20</type>
      </expression>
    </expression>
    <expression>
      <leftSideElement reference="../../../../../
    elements/element[7]"/>
      <rightSideElements/>
      <id>0</id>
      <expression class="constant">
        <value>0</value>
        <type>19</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>shiftm</name>
  <expressions>
    <expression>
      <leftSideElement reference="../../../../../
    elements/element[4]"/>
      <rightSideElements>
        <element reference="../../../../../../
    elements/element[4]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="binaryOperator">
        <expression1 class="variable">
          <element reference="../../../../../../../
         elements/element[4]"/>
          <type>20</type>
        </expression1>
        <expression2 class="constant">
          <value>1</value>
          <type>19</type>
        </expression2>
        <type>16</type>
      </expression>
```

```
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>shiftn</name>
  <expressions>
    <expression>
      <leftSideElement reference="../../../../../
    elements/element[5]"/>
      <rightSideElements>
        <element reference="../../../../../../
       elements/element[5]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="binaryOperator">
        <expression1 class="variable">
          <element reference="../../../../../../../
         elements/element[5]"/>
          <type>20</type>
        </expression1>
        <expression2 class="constant">
          <value>1</value>
          <type>19</type>
        </expression2>
        <type>16</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>reduce</name>
  <expressions>
    <expression>
      <leftSideElement reference="../../../../../
    elements/element[4]"/>
      <rightSideElements>
        <element reference="../../../../../../
    elements/element[4]"/>
        <element reference="../../../../../../
    elements/element[5]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="ternaryOperator">
        <expression1 class="UnaryOperator">
          <expression class="binaryOperator">
            <expression1 class="variable">
              <element reference="../../../../../
                  ../../../../elements/element[4]"/>
              <type>20</type>
            </expression1>
            <expression2 class="variable">
              <element reference="../../../../../
                 ../../../../elements/element[5]"/>
              <type>20</type>
            </expression2>
            <type>23</type>
          </expression>
```

```
      <type>10</type>
    </expression1>
    <expression2 class="binaryOperator">
      <expression1 class="variable">
        <element reference="../../../../../
            ../../../elements/element[4]"/>
        <type>20</type>
      </expression1>
      <expression2 class="variable">
        <element reference="../../../../../
            ../../../elements/element[5]"/>
        <type>20</type>
      </expression2>
      <type>18</type>
    </expression2>
    <expression3 class="variable">
      <element reference="../../../../../
           ../../elements/element[4]"/>
      <type>20</type>
    </expression3>
    <type>14</type>
  </expression>
</expression>
<expression>
  <leftSideElement reference="../../../
     ../../elements/element[5]"/>
  <rightSideElements>
    <element reference="../../../../../../
   elements/element[5]"/>
    <element reference="../../../../../../
   elements/element[4]"/>
  </rightSideElements>
  <id>0</id>
  <expression class="ternaryOperator">
    <expression1 class="UnaryOperator">
      <expression class="binaryOperator">
        <expression1 class="variable">
          <element reference="../../../../../
              ../../../../elements/element[5]"/>
          <type>20</type>
        </expression1>
        <expression2 class="variable">
          <element reference="../../../../../
              ../../../../elements/element[4]"/>
          <type>20</type>
        </expression2>
        <type>25</type>
      </expression>
      <type>10</type>
    </expression1>
    <expression2 class="binaryOperator">
      <expression1 class="variable">
        <element reference="../../../../../
            ../../../elements/element[5]"/>
        <type>20</type>
      </expression1>
      <expression2 class="variable">
```

```
            <element reference="../../../../../
                ../../../elements/element[4]"/>
            <type>20</type>
          </expression2>
          <type>18</type>
        </expression2>
        <expression3 class="variable">
          <element reference="../../../../../
              ../../elements/element[5]"/>
          <type>20</type>
        </expression3>
        <type>14</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>shiftf</name>
  <expressions>
    <expression>
      <leftSideElement reference="../../../../../
    elements/element[7]"/>
      <rightSideElements>
        <element reference="../../../../../../
       elements/element[7]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="binaryOperator">
        <expression1 class="variable">
          <element reference="../../../../../
                ../../elements/element[7]"/>
          <type>20</type>
        </expression1>
        <expression2 class="constant">
          <value>1</value>
          <type>19</type>
        </expression2>
        <type>17</type>
      </expression>
    </expression>
  </expressions>
</sfg>
<sfg>
  <name>outidle</name>
  <expressions>
    <expression>
      <leftSideElement reference="../../../../../
    elements/element[3]"/>
      <rightSideElements/>
      <id>0</id>
      <expression class="constant">
        <value>0</value>
        <type>19</type>
      </expression>
    </expression>
    <expression>
      <leftSideElement reference="../../../../../
```

```
    elements/element[6]"/>
      <rightSideElements>
        <element reference="../../../../../../
    elements/element[4]"/>
        <element reference="../../../../../../
    elements/element[5]"/>
      </rightSideElements>
      <id>0</id>
      <expression class="UnaryOperator">
        <expression class="binaryOperator">
          <expression1 class="UnaryOperator">
            <expression class="binaryOperator">
              <expression1 class="variable">
                <element reference="../../../../
                      ../../../../../../elements/
                                        element[4]">
                  <type>20</type>
                </expression1>
                <expression2 class="constant">
                  <value>0</value>
                  <type>19</type>
                </expression2>
                <type>21</type>
              </expression>
              <type>10</type>
            </expression1>
            <expression2 class="UnaryOperator">
              <expression class="binaryOperator">
                <expression1 class="variable">
                  <element reference="../../../../
                        ../../../../../../elements/
                                        element[5]">
                  <type>20</type>
                </expression1>
                <expression2 class="constant">
                  <value>0</value>
                  <type>19</type>
                </expression2>
                <type>21</type>
              </expression>
              <type>10</type>
            </expression2>
            <type>13</type>
          </expression>
          <type>10</type>
        </expression>
      </expression>
    </expressions>
  </sfg>
  <sfg>
    <name>complete</name>
    <expressions>
      <expression>
        <leftSideElementre ference="../../../
            ../../elements/element[3]"/>
        <rightSideElements>
          <element reference="../../../
```

```
          ../../../elements/element[4]"/>
  <element reference="../../../
          ../../../elements/element[5]"/>
  <element reference="../../../
          ../../../elements/element[7]"/>
</rightSideElements>
<id>0</id>
<expression class="binaryOperator">
  <expression1 class="UnaryOperator">
    <expression class="ternaryOperator">
      <expression1 class="UnaryOperator">
        <expression class="binaryOperator">
          <expression1 class="variable">
               <elementreference="../../../
                     ../../../../../../../../../
                                 elements/
                                 element[4]">
            <type>20</type>
          </expression1>
          <expression2 class="variable">
            <element reference="../../../
                     ../../../../../../../../../
                           elements/
                           element[5]"/>
            <type>20</type>
          </expression2>
          <type>25</type>
        </expression>
        <type>10</type>
      </expression1>
      <expression2 class="variable">
        <element reference="../../../
                     ../../../../../../../
                                elements/
                                element[4]"/>
        <type>20</type>
      </expression2>
      <expression3 class="variable">
        <element reference="../../../
                     ../../../../../../../
                               elements/
                               element[5]"/>
        <type>20</type>
      </expression3>
      <type>14</type>
    </expression>
    <type>10</type>
  </expression1>
  <expression2 class="variable">
    <element reference="../../../
                     ../../../../
                     elements/
                     element[7]"/>
    <type>20</type>
  </expression2>
  <type>15</type>
</expression>
```

```
        </expression>
      </expressions>
    </sfg>
  </sfgs>
  <useModules/>
  <parentModule reference="../.."/>
</datapath>
<fsm>
  <initialState>
    <name>s0</name>
    <transitions>
      <fsmTransition>
        <condition>
          <expression class="constant">
            <value>1</value>
            <type>19</type>
          </expression>
        </condition>
        <usedSfgs>
          <sfg reference="../../../../../../
              datapath/sfgs/sfg"/>
          <sfgreference="../../../../../../
              datapath/sfgs/sfg[6]"/>
        </usedSfgs>
        <nextState>
          <name>s1</name>
          <transitions>
            <fsmTransition>
              <condition>
                <expression class="variable">
                  <element reference="../../../../
                          ../../../../../../datapath/
                                              elements/
                                              element[6]"/
                                          >
                  <type>20</type>
                </expression>
              </condition>
              <usedSfgs>
                <sfg reference="../../../../../../
                    ../../../datapath/sfgs/sfg[7]"/>
              </usedSfgs>
              <nextState>
                <name>s2</name>
                <transitions>
                  <fsmTransition>
                    <condition>
                      <expression class="constant">
                        <value>1</value>
                        <type>19</type>
                      </expression>
                    </condition>
                    <usedSfgs>
                      <sfgreference="../../../../../
                                ../../../../../../../
                                            datapath/
                                        sfgs/sfg[6]/>
```

```
        </usedSfgs>
        <nextState reference="../../.."/>
      </fsmTransition>
    </transitions>
  </nextState>
</fsmTransition>
<fsmTransition>
  <condition>
    <expression class="binaryOperator">
      <expression1 class="bitSelection">
        <element reference="../../../../
                   ../../../../../../../../
                   datapath/elements/
                   element[4]"/>
        <firstBit>0</firstBit>
        <lastBit>0</lastBit>
        <type>26</type>
      </expression1>
      <expression2 class="bitSelection">
        <elementreference="../../../../
                   ../../../../../../../../
                   datapath/elements/
                   element[5]"/>
        <firstBit>0</firstBit>
        <lastBit>0</lastBit>
        <type>26</type>
      </expression2>
      <type>12</type>
    </expression>
  </condition>
  <usedSfgs>
    <sfg reference="../../../../
                   ../../../../../../datapath/
                   sfgs/sfg[4]"/>
    <sfg reference="../../../../
                   ../../../../../../datapath/
                   sfgs/sfg[6]"/>
  </usedSfgs>
  <nextState reference="../../.."/>
</fsmTransition>
<fsmTransition>
  <condition>
    <expression class="binaryOperator">
      <expression1 class="bitSelection">
        <element reference="../../../../
                   ../../../../../../../../
                   datapath/
                   elements/element[4]"/>
        <firstBit>0</firstBit>
        <lastBit>0</lastBit>
        <type>26</type>
      </expression1>
      <expression2 class="UnaryOperator">
        <expression class="bitSelection">
          <element reference="../../../../
              ../../../../../../../../../
                   datapath/
```

```
                    elements/element[5]"/>
              <firstBit>0</firstBit>
              <lastBit>0</lastBit>
              <type>26</type>
            </expression>
            <type>27</type>
          </expression2>
          <type>12</type>
        </expression>
      </condition>
      <usedSfgs>
        <sfg reference="../../../../
                    ../../../../../
                          datapath/
                          sfgs/sfg[3]"/>
        <sfg reference="../../../../
                    ../../../../../
                          datapath/
                          sfgs/sfg[6]"/>
      </usedSfgs>
      <nextState reference="../../.."/>
    </fsmTransition>
    <fsmTransition>
      <condition>
        <expression class="binaryOperator">
          <expression1 class="UnaryOperator">
            <expression class="bitSelection">
              <element reference="../../../../
                    ../../../../../../../../../
                    datapath/
                    elements/element[4]"/>
              <firstBit>0</firstBit>
              <lastBit>0</lastBit>
              <type>26</type>
            </expression>
            <type>27</type>
          </expression1>
          <expression2 class="bitSelection">
            <element reference="../../../../
                    ../../../../../../../../
                    datapath/
                    elements/element[5]"/>
            <firstBit>0</firstBit>
            <lastBit>0</lastBit>
            <type>26</type>
          </expression2>
          <type>12</type>
        </expression>
      </condition>
      <usedSfgs>
        <sfg reference="../../../../
                    ../../../../../
                          datapath/
                          sfgs/sfg[2]"/>
        <sfg reference="../../../../
                    ../../../../../
```

```
                                    datapath/
                                    sfgs/sfg[6]"/>
                </usedSfgs>
                <nextState reference="../../.."/>
              </fsmTransition>
              <fsmTransition>
                <condition>
                  <expression class="constant">
                    <value>1</value>
                    <type>19</type>
                  </expression>
                </condition>
                <usedSfgs>
                  <sfg reference="../../../../
                                  ../../../../../
                                    datapath/
                                    sfgs/sfg[3]"/>
                  <sfg reference="../../../../
                                  ../../../../../
                                    datapath/
                                    sfgs/sfg[2]"/>
                  <sfg reference="../../../../
                                  ../../../../../
                                    datapath/
                                    sfgs/sfg[5]"/>
                  <sfg reference="../../../../
                                  ../../../../../
                                    datapath/
                                    sfgs/sfg[6]"/>
                </usedSfgs>
                <nextState reference="../../.."/>
              </fsmTransition>
            </transitions>
          </nextState>
        </fsmTransition>
      </transitions>
    </initialState>
    <states>
      <fsmState reference="../../initialState"/>
      <fsmState reference="../../
          initialState/transitions/
          fsmTransition/nextState"/>
      <fsmState reference="../../
          initialState/transitions/
          fsmTransition/nextState/transitions/
          fsmTransition/nextState"/>
    </states>
  </fsm>
  <parentModel reference="../../../../../../../"/>
</module>
<parameters>
  <element reference="../../../../elements/element"/>
  <element reference="../../../../elements/element[2]"/>
  <element reference="../../../../elements/element[3]"/>
</parameters>
<name>euclid</name>
</useModule>
```

```
<useModule>
  <module>
    <name>test_euclid</name>
    <datapath>
      <lookupTables/>
      <elements>
        <element>
          <name>m</name>
          <elementType>output</elementType>
          <valueType>ns(5)</valueType>
        </element>
        <element>
          <name>n</name>
          <elementType>output</elementType>
          <valueType>ns(5)</valueType>
        </element>
      </elements>
      <sfgs>
        <sfg>
          <name>always</name>
          <expressions>
            <expression>
              <leftSideElement reference="../../../../../
                  elements/element"/>
              <rightSideElements/>
              <id>0</id>
              <expression class="constant">
                <value>2322</value>
                <type>19</type>
              </expression>
            </expression>
            <expression>
              <leftSideElement reference="../../../../../
                  elements/element[2]"/>
              <rightSideElements/>
              <id>0</id>
              <expression class="constant">
                <value>654</value>
                <type>19</type>
              </expression>
            </expression>
          </expressions>
        </sfg>
      </sfgs>
      <useModules/>
      <parentModule reference="../.."/>
    </datapath>
    <fsm>
      <states>
        <fsmState>
          <name>s0</name>
          <transitions>
            <fsmTransition>
              <condition>
                <expression class="constant">
                  <value>1</value>
                  <type>19</type>
```

```
                        </expression>
                      </condition>
                      <usedSfgs>
                        <sfg reference="../../../../../../../
                          datapath/sfgs/sfg"/>
                      </usedSfgs>
                      <nextState reference="../../.."/>
                    </fsmTransition>
                  </transitions>
                </fsmState>
              </states>
            </fsm>
            <parentModel reference="../../../../../../.."/>
          </module>
          <parameters>
            <element reference="../../../../elements/element"/>
            <element reference="../../../../elements/element[2]"/>
          </parameters>
          <name>test_euclid</name>
        </useModule>
      </useModules>
      <parentModule reference="../.."/>
    </datapath>
    <fsm>
      <states/>
    </fsm>
    <parentModel reference="../../.."/>
  </module>
  <module reference="../module/datapath/
        useModules/useModule/module"/>
  <module reference="../module/datapath/
        useModules/useModule[2]/module"/>
</modules>
<systemModule reference="../modules/module"/>
<ExpressionCounter>0</ExpressionCounter>
</model>
<modules>
  <maaoha.view.serialization.ModuleData>
    <datapathElements>
      <maaoha.view.serialization.ElementData>
        <object class="element" reference="../../../../../
          model/modules/module/datapath/elements/element"/>
        <bounds class=
          "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
          serialization="custom">
          <unserializable-parents/>
          <java.awt.geom.Rectangle2D_-Double>
            <default>
              <height>30.0</height>
              <width>30.0</width>
              <x>570.0</x>
              <y>230.0</y>
            </default>
          </java.awt.geom.Rectangle2D_-Double>
          <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
            <default/>
            <double>570.0</double>
```

```
      <double>230.0</double>
      <double>30.0</double>
      <double>30.0</double>
    </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
  </bounds>
</maaoha.view.serialization.ElementData>
<maaoha.view.serialization.ElementData>
  <object class="element" reference="../../../../../
    model/modules/module/datapath/elements/element[2]"/>
  <bounds class=
    "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Rectangle2D_-Double>
      <default>
        <height>30.0</height>
        <width>30.0</width>
        <x>570.0</x>
        <y>280.0</y>
      </default>
    </java.awt.geom.Rectangle2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      <default/>
      <double>570.0</double>
      <double>280.0</double>
      <double>30.0</double>
      <double>30.0</double>
    </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
  </bounds>
</maaoha.view.serialization.ElementData>
<maaoha.view.serialization.ElementData>
  <object class="element" reference="../../../../../
    model/modules/module/datapath/elements/element[3]"/>
  <bounds class=
    "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Rectangle2D_-Double>
      <default>
        <height>30.0</height>
        <width>30.0</width>
        <x>570.0</x>
        <y>100.0</y>
      </default>
    </java.awt.geom.Rectangle2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      <default/>
      <double>570.0</double>
      <double>100.0</double>
      <double>30.0</double>
      <double>30.0</double>
    </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
  </bounds>
</maaoha.view.serialization.ElementData>
<maaoha.view.serialization.ElementData>
  <object class="useModule" reference="../../../../../
    model/modules/module/datapath/useModules/useModule"/>
```

```
      <bounds class=
        "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
        serialization="custom">
        <unserializable-parents/>
        <java.awt.geom.Rectangle2D_-Double>
          <default>
            <height>80.0</height>
            <width>200.0</width>
            <x>350.0</x>
            <y>100.0</y>
          </default>
        </java.awt.geom.Rectangle2D_-Double>
        <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
          <default/>
          <double>350.0</double>
          <double>100.0</double>
          <double>200.0</double>
          <double>80.0</double>
        </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      </bounds>
    </maaoha.view.serialization.ElementData>
    <maaoha.view.serialization.ElementData>
      <object class="useModule" reference="../../../../../
        model/modules/module/datapath/useModules/useModule[2]"/>
      <bounds class=
        "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
        serialization="custom">
        <unserializable-parents/>
        <java.awt.geom.Rectangle2D_-Double>
          <default>
            <height>80.0</height>
            <width>200.0</width>
            <x>350.0</x>
            <y>230.0</y>
          </default>
        </java.awt.geom.Rectangle2D_-Double>
        <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
          <default/>
          <double>350.0</double>
          <double>230.0</double>
          <double>200.0</double>
          <double>80.0</double>
        </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      </bounds>
    </maaoha.view.serialization.ElementData>
  </datapathElements>
  <controllerElements/>
  <edgeLabelsElements/>
</maaoha.view.serialization.ModuleData>
<maaoha.view.serialization.ModuleData>
  <datapathElements>
    <maaoha.view.serialization.ElementData>
      <object class="element" reference="../../../../../
        model/modules/module/datapath/useModules/useModule/module/
        datapath/elements/element"/>
      <bounds class=
```

```
    "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Rectangle2D_-Double>
      <default>
        <height>20.0</height>
        <width>80.0</width>
        <x>0.0</x>
        <y>100.0</y>
      </default>
    </java.awt.geom.Rectangle2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      <default/>
      <double>0.0</double>
      <double>100.0</double>
      <double>80.0</double>
      <double>20.0</double>
    </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
  </bounds>
</maaoha.view.serialization.ElementData>
<maaoha.view.serialization.ElementData>
  <object class="element" reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    datapath/elements/element[2]"/>
  <bounds class=
    "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Rectangle2D_-Double>
      <default>
        <height>20.0</height>
        <width>80.0</width>
        <x>0.0</x>
        <y>170.0</y>
      </default>
    </java.awt.geom.Rectangle2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      <default/>
      <double>0.0</double>
      <double>170.0</double>
      <double>80.0</double>
      <double>20.0</double>
    </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
  </bounds>
</maaoha.view.serialization.ElementData>
<maaoha.view.serialization.ElementData>
  <object class="element" reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    datapath/elements/element[3]"/>
  <bounds class=
    "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Rectangle2D_-Double>
      <default>
        <height>20.0</height>
        <width>80.0</width>
```

```
        <x>700.0</x>
        <y>100.0</y>
      </default>
    </java.awt.geom.Rectangle2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      <default/>
      <double>700.0</double>
      <double>100.0</double>
      <double>80.0</double>
      <double>20.0</double>
    </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
  </bounds>
</maaoha.view.serialization.ElementData>
<maaoha.view.serialization.ElementData>
  <object class="element" reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    datapath/elements/element[4]"/>
  <bounds class=
    "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Rectangle2D_-Double>
      <default>
        <height>40.0</height>
        <width>40.0</width>
        <x>350.0</x>
        <y>100.0</y>
      </default>
    </java.awt.geom.Rectangle2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      <default/>
      <double>350.0</double>
      <double>100.0</double>
      <double>40.0</double>
      <double>40.0</double>
    </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
  </bounds>
</maaoha.view.serialization.ElementData>
<maaoha.view.serialization.ElementData>
  <object class="element" reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    datapath/elements/element[5]"/>
  <bounds class=
    "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Rectangle2D_-Double>
      <default>
        <height>40.0</height>
        <width>40.0</width>
        <x>350.0</x>
        <y>190.0</y>
      </default>
    </java.awt.geom.Rectangle2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      <default/>
      <double>350.0</double>
```

```
        <double>190.0</double>
        <double>40.0</double>
        <double>40.0</double>
      </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
    </bounds>
  </maaoha.view.serialization.ElementData>
  <maaoha.view.serialization.ElementData>
    <object class="element" reference="../../../../../
      model/modules/module/datapath/useModules/useModule/module/
      datapath/elements/element[6]"/>
    <bounds class=
      "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
      serialization="custom">
      <unserializable-parents/>
      <java.awt.geom.Rectangle2D_-Double>
        <default>
          <height>40.0</height>
          <width>40.0</width>
          <x>350.0</x>
          <y>280.0</y>
        </default>
      </java.awt.geom.Rectangle2D_-Double>
      <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
        <default/>
        <double>350.0</double>
        <double>280.0</double>
        <double>40.0</double>
        <double>40.0</double>
      </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
    </bounds>
  </maaoha.view.serialization.ElementData>
  <maaoha.view.serialization.ElementData>
    <object class="element" reference="../../../../../
      model/modules/module/datapath/useModules/useModule/module/
      datapath/elements/element[7]"/>
    <bounds class=
      "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
      serialization="custom">
      <unserializable-parents/>
      <java.awt.geom.Rectangle2D_-Double>
        <default>
          <height>40.0</height>
          <width>40.0</width>
          <x>350.0</x>
          <y>370.0</y>
        </default>
      </java.awt.geom.Rectangle2D_-Double>
      <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
        <default/>
        <double>350.0</double>
        <double>370.0</double>
        <double>40.0</double>
        <double>40.0</double>
      </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
    </bounds>
  </maaoha.view.serialization.ElementData>
</datapathElements>
```

```
<controllerElements>
  <maaoha.view.serialization.ElementData>
    <object class="fsmState" reference="../../../../../
      model/modules/module/datapath/useModules/useModule/module/
      fsm/initialState"/>
    <bounds class=
      "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
      serialization="custom">
      <unserializable-parents/>
      <java.awt.geom.Rectangle2D_-Double>
        <default>
          <height>50.0</height>
          <width>50.0</width>
          <x>100.0</x>
          <y>50.0</y>
        </default>
      </java.awt.geom.Rectangle2D_-Double>
      <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
        <default/>
        <double>100.0</double>
        <double>50.0</double>
        <double>50.0</double>
        <double>50.0</double>
      </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
    </bounds>
  </maaoha.view.serialization.ElementData>
  <maaoha.view.serialization.ElementData>
    <object class="fsmState" reference="../../../../../
      model/modules/module/datapath/useModules/useModule/module/
      fsm/initialState/transitions/fsmTransition/nextState"/>
    <bounds class=
      "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
      serialization="custom">
      <unserializable-parents/>
      <java.awt.geom.Rectangle2D_-Double>
        <default>
          <height>50.0</height>
          <width>50.0</width>
          <x>600.0</x>
          <y>50.0</y>
        </default>
      </java.awt.geom.Rectangle2D_-Double>
      <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
        <default/>
        <double>600.0</double>
        <double>50.0</double>
        <double>50.0</double>
        <double>50.0</double>
      </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
    </bounds>
  </maaoha.view.serialization.ElementData>
  <maaoha.view.serialization.ElementData>
    <object class="fsmState" reference="../../../../../
      model/modules/module/datapath/useModules/useModule/module/
      fsm/initialState/transitions/fsmTransition/nextState/
      transitions/fsmTransition/nextState"/>
    <bounds class=
```

```
      "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
      serialization="custom">
      <unserializable-parents/>
      <java.awt.geom.Rectangle2D_-Double>
        <default>
          <height>50.0</height>
          <width>50.0</width>
          <x>100.0</x>
          <y>250.0</y>
        </default>
      </java.awt.geom.Rectangle2D_-Double>
      <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
        <default/>
        <double>100.0</double>
        <double>250.0</double>
        <double>50.0</double>
        <double>50.0</double>
      </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
    </bounds>
  </maaoha.view.serialization.ElementData>
</controllerElements>
<edgeLabelsElements>
  <maaoha.view.serialization.EdgeData>
    <transition reference="../../../../../
      model/modules/module/datapath/useModules/useModule/module/
      fsm/initialState/transitions/fsmTransition"/>
    <labelPosition class=
      "org.jgraph.graph.AttributeMap$SerializablePoint2D"
      serialization="custom">
      <unserializable-parents/>
      <java.awt.geom.Point2D_-Double>
        <default>
          <x>375.0</x>
          <y>0.0</y>
        </default>
      </java.awt.geom.Point2D_-Double>
      <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
        <default/>
        <double>375.0</double>
        <double>0.0</double>
      </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
    </labelPosition>
  </maaoha.view.serialization.EdgeData>
  <maaoha.view.serialization.EdgeData>
    <transition reference="../../../../../
      model/modules/module/datapath/useModules/useModule/module/
      fsm/initialState/transitions/fsmTransition/nextState/
      transitions/fsmTransition"/>
    <labelPosition class=
      "org.jgraph.graph.AttributeMap$SerializablePoint2D"
      serialization="custom">
      <unserializable-parents/>
      <java.awt.geom.Point2D_-Double>
        <default>
          <x>375.0</x>
          <y>0.0</y>
        </default>
```

```
    </java.awt.geom.Point2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
      <default/>
      <double>375.0</double>
      <double>0.0</double>
    </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
  </labelPosition>
</maaoha.view.serialization.EdgeData>
<maaoha.view.serialization.EdgeData>
  <transition reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    fsm/initialState/transitions/fsmTransition/nextState/
    transitions/fsmTransition[2]"/>
  <labelPosition class=
    "org.jgraph.graph.AttributeMap$SerializablePoint2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Point2D_-Double>
      <default>
        <x>375.0</x>
        <y>0.0</y>
      </default>
    </java.awt.geom.Point2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
      <default/>
      <double>375.0</double>
      <double>0.0</double>
    </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
  </labelPosition>
</maaoha.view.serialization.EdgeData>
<maaoha.view.serialization.EdgeData>
  <transition reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    fsm/initialState/transitions/fsmTransition/nextState/
    transitions/fsmTransition[3]"/>
  <labelPosition class=
    "org.jgraph.graph.AttributeMap$SerializablePoint2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Point2D_-Double>
      <default>
        <x>255.0</x>
        <y>0.0</y>
      </default>
    </java.awt.geom.Point2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
      <default/>
      <double>255.0</double>
      <double>0.0</double>
    </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
  </labelPosition>
</maaoha.view.serialization.EdgeData>
<maaoha.view.serialization.EdgeData>
  <transition reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    fsm/initialState/transitions/fsmTransition/nextState/
    transitions/fsmTransition[4]"/>
```

```
  <labelPosition class=
    "org.jgraph.graph.AttributeMap$SerializablePoint2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Point2D_-Double>
      <default>
        <x>135.0</x>
        <y>0.0</y>
      </default>
    </java.awt.geom.Point2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
      <default/>
      <double>135.0</double>
      <double>0.0</double>
    </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
  </labelPosition>
</maaoha.view.serialization.EdgeData>
<maaoha.view.serialization.EdgeData>
  <transition reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    fsm/initialState/transitions/fsmTransition/nextState/
    transitions/fsmTransition[5]"/>
  <labelPosition class=
    "org.jgraph.graph.AttributeMap$SerializablePoint2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Point2D_-Double>
      <default>
        <x>15.0</x>
        <y>0.0</y>
      </default>
    </java.awt.geom.Point2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
      <default/>
      <double>15.0</double>
      <double>0.0</double>
    </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
  </labelPosition>
</maaoha.view.serialization.EdgeData>
<maaoha.view.serialization.EdgeData>
  <transition reference="../../../../../
    model/modules/module/datapath/useModules/useModule/module/
    fsm/initialState/transitions/fsmTransition/nextState/
    transitions/fsmTransition/nextState/transitions/
    fsmTransition"/>
  <labelPosition class=
    "org.jgraph.graph.AttributeMap$SerializablePoint2D"
    serialization="custom">
    <unserializable-parents/>
    <java.awt.geom.Point2D_-Double>
      <default>
        <x>375.0</x>
        <y>0.0</y>
      </default>
    </java.awt.geom.Point2D_-Double>
    <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
      <default/>
```

```
          <double>375.0</double>
          <double>0.0</double>
        </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
      </labelPosition>
    </maaoha.view.serialization.EdgeData>
  </edgeLabelsElements>
</maaoha.view.serialization.ModuleData>
<maaoha.view.serialization.ModuleData>
  <datapathElements>
    <maaoha.view.serialization.ElementData>
      <object class="element" reference="../../../../../
        model/modules/module/datapath/useModules/useModule[2]/module
        /datapath/elements/element"/>
      <bounds class=
        "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
        serialization="custom">
        <unserializable-parents/>
        <java.awt.geom.Rectangle2D_-Double>
          <default>
            <height>20.0</height>
            <width>80.0</width>
            <x>700.0</x>
            <y>100.0</y>
          </default>
        </java.awt.geom.Rectangle2D_-Double>
        <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
          <default/>
          <double>700.0</double>
          <double>100.0</double>
          <double>80.0</double>
          <double>20.0</double>
        </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
      </bounds>
    </maaoha.view.serialization.ElementData>
    <maaoha.view.serialization.ElementData>
      <object class="element" reference="../../../../../
        model/modules/module/datapath/useModules/useModule[2]/module
        /datapath/elements/element[2]"/>
      <bounds class=
        "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
        serialization="custom">
        <unserializable-parents/>
        <java.awt.geom.Rectangle2D_-Double>
          <default>
            <height>20.0</height>
            <width>80.0</width>
            <x>700.0</x>
            <y>170.0</y>
          </default>
        </java.awt.geom.Rectangle2D_-Double>
        <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
          <default/>
          <double>700.0</double>
          <double>170.0</double>
          <double>80.0</double>
          <double>20.0</double>
        </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
```

```
          </bounds>
        </maaoha.view.serialization.ElementData>
      </datapathElements>
      <controllerElements>
        <maaoha.view.serialization.ElementData>
          <object class="fsmState" reference="../../../../../
            model/modules/module/datapath/useModules/useModule[2]/module
            /fsm/states/fsmState"/>
          <bounds class=
            "org.jgraph.graph.AttributeMap$SerializableRectangle2D"
            serialization="custom">
            <unserializable-parents/>
            <java.awt.geom.Rectangle2D_-Double>
              <default>
                <height>50.0</height>
                <width>50.0</width>
                <x>100.0</x>
                <y>50.0</y>
              </default>
            </java.awt.geom.Rectangle2D_-Double>
            <org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
              <default/>
              <double>100.0</double>
              <double>50.0</double>
              <double>50.0</double>
              <double>50.0</double>
            </org.jgraph.graph.AttributeMap_-SerializableRectangle2D>
          </bounds>
        </maaoha.view.serialization.ElementData>
      </controllerElements>
      <edgeLabelsElements>
        <maaoha.view.serialization.EdgeData>
          <transition reference="../../../../../
            model/modules/module/datapath/useModules/useModule[2]/module
            /fsm/states/fsmState/transitions/fsmTransition"/>
          <labelPosition class=
            "org.jgraph.graph.AttributeMap$SerializablePoint2D"
            serialization="custom">
            <unserializable-parents/>
            <java.awt.geom.Point2D_-Double>
              <default>
                <x>375.0</x>
                <y>0.0</y>
              </default>
            </java.awt.geom.Point2D_-Double>
            <org.jgraph.graph.AttributeMap_-SerializablePoint2D>
              <default/>
              <double>375.0</double>
              <double>0.0</double>
            </org.jgraph.graph.AttributeMap_-SerializablePoint2D>
          </labelPosition>
        </maaoha.view.serialization.EdgeData>
      </edgeLabelsElements>
    </maaoha.view.serialization.ModuleData>
  </modules>
</maaoha.view.serialization.ModelData>
```

A.3 NuSMV sources

A.3.1 Euclid algorithm

```
MODULE main

VAR

sig_m : word[16];
sig_n : word[16];
sig_gcd : word[16];

um_euclid : mod_euclid(sig_m, sig_n, sig_gcd);
um_test_euclid : mod_test_euclid(sig_m, sig_n);

MODULE mod_euclid(inp_m_in, inp_n_in, out_gcd)

VAR

ctl_euclid :  {st_s0, st_s1, st_s2};

reg_m : word[16];
reg_n : word[16];
reg_done : word[1];
reg_factor : word[16];


sfg_init : boolean;
sfg_shiftm : boolean;
sfg_shiftn : boolean;
sfg_reduce : boolean;
sfg_shiftf : boolean;
sfg_outidle : boolean;
sfg_complete : boolean;

INIT
ctl_euclid = st_s0;
TRANS
next(ctl_euclid) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : st_s2;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : st_s1;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : st_s2;
      1 : ctl_euclid;
   esac

INIT
out_gcd = 0d16_0;
```

```
TRANS
next(out_gcd) =
   case
      next(sfg_outidle) = TRUE & (0d1_1) > 0d1_0 : 0d15_0 :: (0d1_0);
      next(sfg_complete) = TRUE & ((word1(reg_m > reg_n))) > 0d1_0 :
            ((((reg_m))) << ((reg_factor)[3:0]))[15:0];
      next(sfg_complete) = TRUE & (0d1_1) > 0d1_0 :
            ((((reg_n))) << ((reg_factor)[3:0]))[15:0];
      1 : out_gcd;
   esac

INIT
reg_m = 0d16_0;
TRANS
next(reg_m) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : next(inp_m_in);
      next(sfg_shiftm) = TRUE & (0d1_1) > 0d1_0 :
            (((reg_m)) >> ((0d1_1)))[15:0];
      next(sfg_reduce) = TRUE & ((word1(reg_m >= reg_n))) > 0d1_0 :
            reg_m - reg_n;
      next(sfg_reduce) = TRUE & (0d1_1) > 0d1_0 : reg_m;
      1 : reg_m;
   esac

INIT
reg_n = 0d16_0;
TRANS
next(reg_n) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : next(inp_n_in);
      next(sfg_shiftn) = TRUE & (0d1_1) > 0d1_0 :
            (((reg_n)) >> ((0d1_1)))[15:0];
      next(sfg_reduce) = TRUE & ((word1(reg_n > reg_m))) > 0d1_0 :
            reg_n - reg_m;
      next(sfg_reduce) = TRUE & (0d1_1) > 0d1_0 : reg_n;
      1 : reg_n;
   esac

INIT
reg_done = 0d1_0;
TRANS
next(reg_done) =
   case
      next(sfg_outidle) = TRUE & (0d1_1) > 0d1_0 :
            ((word1(reg_m = (( 0d15_0 ) :: 0d1_0))) |
            (word1(reg_n = (( 0d15_0 ) :: 0d1_0))));
      1 : reg_done;
   esac

INIT
reg_factor = 0d16_0;
TRANS
next(reg_factor) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : 0d15_0 :: (0d1_0);
      next(sfg_shiftf) = TRUE & (0d1_1) > 0d1_0 :
```

```
            reg_factor + (( 0d15_0 ) :: 0d1_1);
      1 : reg_factor;
   esac

INIT
sfg_init = FALSE;
TRANS
next(sfg_init) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftm = FALSE;
TRANS
next(sfg_shiftm) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftn = FALSE;
TRANS
next(sfg_shiftn) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_reduce = FALSE;
TRANS
next(sfg_reduce) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
```

```
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftf = FALSE;
TRANS
next(sfg_shiftf) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_outidle = FALSE;
TRANS
next(sfg_outidle) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac

INIT
sfg_complete = FALSE;
TRANS
next(sfg_complete) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac


MODULE mod_test_euclid(out_m, out_n)

VAR
```

```
ctl_test_euclid :  {st_s0};


sfg_always : boolean;

INIT
ctl_test_euclid = st_s0;
TRANS
next(ctl_test_euclid) =
   case
      ctl_test_euclid = st_s0 & (0d1_1) > 0d1_0 : st_s0;
      1 : ctl_test_euclid;
   esac

INIT
out_m = 0d16_0;
TRANS
next(out_m) =
   case
      next(sfg_always) = TRUE & (0d1_1) > 0d1_0 : 0d4_0 :: (0d12_2322);
      1 : out_m;
   esac

INIT
out_n = 0d16_0;
TRANS
next(out_n) =
   case
      next(sfg_always) = TRUE & (0d1_1) > 0d1_0 : 0d6_0 :: (0d10_654);
      1 : out_n;
   esac

INIT
sfg_always = FALSE;
TRANS
next(sfg_always) =
   case
      ctl_test_euclid = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac
```

A.3.2 Simplified DES

```
MODULE main

VAR

sig_n : word[8];
sig_o : word[8];
sig_oo : word[8];
sig_k : word[10];
sig_d : word[1];
sig_dd : word[1];
```

```
sig_s : word[1];

um_test_encrypt : mod_test_encrypt(sig_n, sig_k);
um_encrypt : mod_encrypt(sig_n, sig_k, sig_o, sig_d);
um_decrypt : mod_decrypt(sig_o, sig_k, sig_d, sig_oo, sig_dd);


INIT
sig_s = 0d1_0;
TRANS
next(sig_s) =
   case
      1 : sig_s;
   esac


MODULE mod_encrypt(inp_in_port, inp_in_key, out_out_port, out_done)

VAR

ctl_encrypt :  {st_s0, st_s1, st_s2, st_s3, st_s4, st_s5, st_s6, st_s7,
st_s8, st_s9};

reg_input : word[8];
reg_k1 : word[8];
reg_k2 : word[8];
reg_res : word[8];
reg_fk : word[8];
reg_sw : word[8];
reg_key : word[10];
reg_left : word[4];
reg_right : word[4];
sig_ls1 : word[10];
sig_ls2 : word[10];
sig_pk : word[10];
sig_p4 : word[4];
sig_fkl : word[4];
sig_ep : word[8];
sig_s0r : word[2];
sig_s0c : word[2];
sig_s1r : word[2];
sig_s1c : word[2];
sig_first : word[2];
sig_second : word[2];


sfg_ini : boolean;
sfg_keys : boolean;
sfg_lr : boolean;
sfg_ip : boolean;
sfg_slook : boolean;
sfg_rl : boolean;
sfg_secondf : boolean;
sfg_setout : boolean;

INIT
ctl_encrypt = st_s0;
```

```
TRANS
next(ctl_encrypt) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : st_s1;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : st_s2;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : st_s3;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : st_s4;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : st_s5;
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : st_s6;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : st_s7;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : st_s8;
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : st_s9;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : st_s9;
      1 : ctl_encrypt;
   esac

INIT
out_out_port = 0d8_0;
TRANS
next(out_out_port) =
   case
      next(sfg_ini) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_lr) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_ip) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_rl) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_secondf) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_setout) = TRUE & (0d1_1) > 0d1_0 : reg_fk[4:4] ::
            reg_fk[7:7] :: reg_fk[5:5] :: reg_fk[3:3] :: reg_fk[1:1] ::
            reg_fk[6:6] :: reg_fk[0:0] :: reg_fk[2:2];
      1 : out_out_port;
   esac

INIT
out_done = 0d1_0;
TRANS
next(out_done) =
   case
      next(sfg_ini) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_lr) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_ip) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_rl) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_secondf) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_setout) = TRUE & (0d1_1) > 0d1_0 : 0d1_1;
      1 : out_done;
   esac

INIT
reg_input = 0d8_0;
TRANS
next(reg_input) =
   case
      next(sfg_ini) = TRUE & (0d1_1) > 0d1_0 : next(inp_in_port);
      1 : reg_input;
```

```
   esac

INIT
reg_k1 = 0d8_0;
TRANS
next(reg_k1) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : sig_ls1[4:4] ::
            sig_ls1[7:7] :: sig_ls1[3:3] :: sig_ls1[6:6] ::
            sig_ls1[2:2] :: sig_ls1[5:5] :: sig_ls1[0:0] ::
            sig_ls1[1:1];
      1 : reg_k1;
   esac

INIT
reg_k2 = 0d8_0;
TRANS
next(reg_k2) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : sig_ls2[4:4] ::
            sig_ls2[7:7] :: sig_ls2[3:3] :: sig_ls2[6:6] ::
            sig_ls2[2:2] :: sig_ls2[5:5] :: sig_ls2[0:0] ::
            sig_ls2[1:1];
      1 : reg_k2;
   esac

INIT
reg_res = 0d8_0;
TRANS
next(reg_res) =
   case
      next(sfg_ip) = TRUE & (0d1_1) > 0d1_0 : next(sig_ep) xor reg_k1;
      next(sfg_secondf) = TRUE & (0d1_1) > 0d1_0 : n
            ext(sig_ep) xor reg_k2;
      1 : reg_res;
   esac

INIT
reg_fk = 0d8_0;
TRANS
next(reg_fk) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 :
            next(sig_fkl) :: reg_right;
      1 : reg_fk;
   esac

INIT
reg_sw = 0d8_0;
TRANS
next(reg_sw) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 :
            reg_right :: next(sig_fkl);
      1 : reg_sw;
   esac
```

```
INIT
reg_key = 0d10_0;
TRANS
next(reg_key) =
   case
      next(sfg_ini) = TRUE & (0d1_1) > 0d1_0 : next(inp_in_key);
      1 : reg_key;
   esac

INIT
reg_left = 0d4_0;
TRANS
next(reg_left) =
   case
      next(sfg_lr) = TRUE & (0d1_1) > 0d1_0 : reg_input[6:6] ::
            reg_input[2:2] :: reg_input[5:5] :: reg_input[7:7];
      next(sfg_rl) = TRUE & (0d1_1) > 0d1_0 : reg_sw[7:7] ::
            reg_sw[6:6] :: reg_sw[5:5] :: reg_sw[4:4];
      1 : reg_left;
   esac

INIT
reg_right = 0d4_0;
TRANS
next(reg_right) =
   case
      next(sfg_lr) = TRUE & (0d1_1) > 0d1_0 : reg_input[4:4] ::
            reg_input[0:0] :: reg_input[3:3] :: reg_input[1:1];
      next(sfg_rl) = TRUE & (0d1_1) > 0d1_0 : reg_sw[3:3] ::
            reg_sw[2:2] :: reg_sw[1:1] :: reg_sw[0:0];
      1 : reg_right;
   esac

INIT
sig_ls1 = 0d10_0;
TRANS
next(sig_ls1) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : sig_pk[8:8] ::
            sig_pk[7:7] :: sig_pk[6:6] :: sig_pk[5:5] :: sig_pk[9:9] ::
            sig_pk[3:3] :: sig_pk[2:2] :: sig_pk[1:1] :: sig_pk[0:0] ::
            sig_pk[4:4];
      1 : sig_ls1;
   esac

INIT
sig_ls2 = 0d10_0;
TRANS
next(sig_ls2) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : sig_ls1[7:7] ::
            sig_ls1[6:6] :: sig_ls1[5:5] :: sig_ls1[9:9] ::
            sig_ls1[8:8] :: sig_ls1[2:2] :: sig_ls1[1:1] ::
            sig_ls1[0:0] :: sig_ls1[4:4] :: sig_ls1[3:3];
      1 : sig_ls2;
   esac
```

```
INIT
sig_pk = 0d10_0;
TRANS
next(sig_pk) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : reg_key[5:5] ::
            reg_key[7:7] :: reg_key[8:8] :: reg_key[0:0] ::
            reg_key[9:9] :: reg_key[3:3] :: reg_key[6:6] ::
            reg_key[1:1] :: reg_key[4:4] :: reg_key[2:2];
      1 : sig_pk;
   esac

INIT
sig_p4 = 0d4_0;
TRANS
next(sig_p4) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : sig_first[0:0] ::
            sig_second[0:0] :: sig_second[1:1] :: sig_first[1:1];
      1 : sig_p4;
   esac

INIT
sig_fkl = 0d4_0;
TRANS
next(sig_fkl) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 :
            reg_left xor next(sig_p4);
      1 : sig_fkl;
   esac

INIT
sig_ep = 0d8_0;
TRANS
next(sig_ep) =
   case
      next(sfg_ip) = TRUE & (0d1_1) > 0d1_0 : reg_right[0:0] ::
            reg_right[3:3] :: reg_right[2:2] :: reg_right[1:1] ::
            reg_right[2:2] :: reg_right[1:1] :: reg_right[0:0] ::
            reg_right[3:3];
      next(sfg_secondf) = TRUE & (0d1_1) > 0d1_0 : reg_right[0:0] ::
            reg_right[3:3] :: reg_right[2:2] :: reg_right[1:1] ::
            reg_right[2:2] :: reg_right[1:1] :: reg_right[0:0] ::
            reg_right[3:3];
      1 : sig_ep;
   esac

INIT
sig_s0r = 0d2_0;
TRANS
next(sig_s0r) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : reg_res[7:7] ::
            reg_res[4:4];
      1 : sig_s0r;
   esac
```

```
INIT
sig_s0c = 0d2_0;
TRANS
next(sig_s0c) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : reg_res[6:6] ::
            reg_res[5:5];
      1 : sig_s0c;
   esac

INIT
sig_s1r = 0d2_0;
TRANS
next(sig_s1r) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : reg_res[0:0] ::
            reg_res[3:3];
      1 : sig_s1r;
   esac

INIT
sig_s1c = 0d2_0;
TRANS
next(sig_s1c) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : reg_res[2:2] ::
            reg_res[1:1];
      1 : sig_s1c;
   esac

INIT
sig_first = 0d2_0;
TRANS
next(sig_first) =
   case
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_0))) >
            0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_1))) >
            0d1_0 : 0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s0c) = 0d2_2)) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s0c) = 0d2_3)) > 0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_0))) >
            0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_1))) >
            0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s0c) = 0d2_2)) > 0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s0c) = 0d2_3)) > 0d1_0 : 0d1_0 ::
            (0d1_0);
```

```
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = 0d2_2)) &
            word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_0))) > 0d1_0 :
            0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = 0d2_2)) &
            word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_1))) > 0d1_0 :
            0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = 0d2_2)) &
            word1(next(sig_s0c) = 0d2_2)) > 0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = 0d2_2)) &
            word1(next(sig_s0c) = 0d2_3)) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & (word1(next(sig_s0c) = (( 0d1_0 ) ::
            0d1_0))) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & (word1(next(sig_s0c) = (( 0d1_0 ) ::
            0d1_1))) > 0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & (word1(next(sig_s0c) = 0d2_2)) > 0d1_0 :
            0d2_3;
      next(sfg_slook) = TRUE & (word1(next(sig_s0c) = 0d2_3)) > 0d1_0 :
            0d2_2;
      1 : sig_first;
   esac

INIT
sig_second = 0d2_0;
TRANS
next(sig_second) =
   case
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_0))) >
            0d1_0 : 0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_1))) >
            0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s1c) = 0d2_2)) > 0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s1c) = 0d2_3)) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_0))) >
            0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_1))) >
            0d1_0 : 0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s1c) = 0d2_2)) > 0d1_0 : 0d1_0 ::
            (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s1c) = 0d2_3)) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = 0d2_2)) &
            word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_0))) > 0d1_0 :
            0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = 0d2_2)) &
            word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_1))) > 0d1_0 :
            0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = 0d2_2)) &
            word1(next(sig_s1c) = 0d2_2)) > 0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = 0d2_2)) &
            word1(next(sig_s1c) = 0d2_3)) > 0d1_0 : 0d1_0 :: (0d1_0);
```

```
      next(sfg_slook) = TRUE & (word1(next(sig_s1c) = (( 0d1_0 ) ::
            0d1_0))) > 0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & (word1(next(sig_s1c) = (( 0d1_0 ) ::
            0d1_1))) > 0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & (word1(next(sig_s1c) = 0d2_2)) > 0d1_0 :
            0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & (word1(next(sig_s1c) = 0d2_3)) > 0d1_0 :
            0d2_3;
      1 : sig_second;
   esac

INIT
sfg_ini = FALSE;
TRANS
next(sfg_ini) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_keys = FALSE;
TRANS
next(sfg_keys) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_lr = FALSE;
TRANS
next(sfg_lr) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
```

```
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_ip = FALSE;
TRANS
next(sfg_ip) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_slook = FALSE;
TRANS
next(sfg_slook) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_rl = FALSE;
TRANS
next(sfg_rl) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
```

```
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_secondf = FALSE;
TRANS
next(sfg_secondf) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_setout = FALSE;
TRANS
next(sfg_setout) =
   case
      ctl_encrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_encrypt = st_s8 & (0d1_1) > 0d1_0 : TRUE;
      ctl_encrypt = st_s9 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac


MODULE mod_decrypt(inp_in_port, inp_in_key, inp_in_start,
                        out_out_port, out_done)

VAR

ctl_decrypt :  {st_s0, st_s1, st_s2, st_s3, st_s4, st_s5, st_s6, st_s7,
                   st_s8, st_s9};

reg_input : word[8];
reg_k1 : word[8];
reg_k2 : word[8];
reg_res : word[8];
reg_fk : word[8];
reg_sw : word[8];
reg_key : word[10];
```

```
reg_left : word[4];
reg_right : word[4];
reg_start : word[1];
sig_ls1 : word[10];
sig_ls2 : word[10];
sig_pk : word[10];
sig_p4 : word[4];
sig_fk1 : word[4];
sig_ep : word[8];
sig_s0r : word[2];
sig_s0c : word[2];
sig_s1r : word[2];
sig_s1c : word[2];
sig_first : word[2];
sig_second : word[2];


sfg_pre : boolean;
sfg_ini : boolean;
sfg_keys : boolean;
sfg_lr : boolean;
sfg_ip : boolean;
sfg_slook : boolean;
sfg_rl : boolean;
sfg_secondf : boolean;
sfg_setout : boolean;

INIT
ctl_decrypt = st_s0;
TRANS
next(ctl_decrypt) =
   case
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : st_s1;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : st_s0;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : st_s2;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : st_s3;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : st_s4;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : st_s5;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : st_s6;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : st_s7;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : st_s8;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : st_s9;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : st_s9;
      1 : ctl_decrypt;
   esac

INIT
out_out_port = 0d8_0;
TRANS
next(out_out_port) =
   case
      next(sfg_pre) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_ini) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_lr) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_ip) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
```

```
      next(sfg_rl) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_secondf) = TRUE & (0d1_1) > 0d1_0 : 0d7_0 :: (0d1_0);
      next(sfg_setout) = TRUE & (0d1_1) > 0d1_0 : reg_fk[4:4] ::
            reg_fk[7:7] :: reg_fk[5:5] :: reg_fk[3:3] :: reg_fk[1:1] ::
            reg_fk[6:6] :: reg_fk[0:0] :: reg_fk[2:2];
      1 : out_out_port;
   esac

INIT
out_done = 0d1_0;
TRANS
next(out_done) =
   case
      next(sfg_pre) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_ini) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_lr) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_ip) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_rl) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_secondf) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_setout) = TRUE & (0d1_1) > 0d1_0 : 0d1_1;
      1 : out_done;
   esac

INIT
reg_input = 0d8_0;
TRANS
next(reg_input) =
   case
      next(sfg_ini) = TRUE & (0d1_1) > 0d1_0 : next(inp_in_port);
      1 : reg_input;
   esac

INIT
reg_k1 = 0d8_0;
TRANS
next(reg_k1) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : sig_ls1[4:4] ::
            sig_ls1[7:7] :: sig_ls1[3:3] :: sig_ls1[6:6] ::
            sig_ls1[2:2] :: sig_ls1[5:5] :: sig_ls1[0:0] ::
            sig_ls1[1:1];
      1 : reg_k1;
   esac

INIT
reg_k2 = 0d8_0;
TRANS
next(reg_k2) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : sig_ls2[4:4] ::
            sig_ls2[7:7] :: sig_ls2[3:3] :: sig_ls2[6:6] ::
            sig_ls2[2:2] :: sig_ls2[5:5] :: sig_ls2[0:0] ::
            sig_ls2[1:1];
      1 : reg_k2;
   esac
```

```
INIT
reg_res = 0d8_0;
TRANS
next(reg_res) =
   case
      next(sfg_ip) = TRUE & (0d1_1) > 0d1_0 : next(sig_ep) xor reg_k2;
      next(sfg_secondf) = TRUE & (0d1_1) > 0d1_0 : next(sig_ep) xor
            reg_k1;
      1 : reg_res;
   esac

INIT
reg_fk = 0d8_0;
TRANS
next(reg_fk) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 :
            next(sig_fkl) :: reg_right;
      1 : reg_fk;
   esac

INIT
reg_sw = 0d8_0;
TRANS
next(reg_sw) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 :
            reg_right :: next(sig_fkl);
      1 : reg_sw;
   esac

INIT
reg_key = 0d10_0;
TRANS
next(reg_key) =
   case
      next(sfg_ini) = TRUE & (0d1_1) > 0d1_0 : next(inp_in_key);
      1 : reg_key;
   esac

INIT
reg_left = 0d4_0;
TRANS
next(reg_left) =
   case
      next(sfg_lr) = TRUE & (0d1_1) > 0d1_0 : reg_input[6:6] ::
            reg_input[2:2] :: reg_input[5:5] :: reg_input[7:7];
      next(sfg_rl) = TRUE & (0d1_1) > 0d1_0 : reg_sw[7:7] ::
            reg_sw[6:6] :: reg_sw[5:5] :: reg_sw[4:4];
      1 : reg_left;
   esac

INIT
reg_right = 0d4_0;
TRANS
next(reg_right) =
```

```
   case
      next(sfg_lr) = TRUE & (0d1_1) > 0d1_0 : reg_input[4:4] ::
            reg_input[0:0] :: reg_input[3:3] :: reg_input[1:1];
      next(sfg_rl) = TRUE & (0d1_1) > 0d1_0 : reg_sw[3:3] ::
            reg_sw[2:2] :: reg_sw[1:1] :: reg_sw[0:0];
      1 : reg_right;
   esac

INIT
reg_start = 0d1_0;
TRANS
next(reg_start) =
   case
      next(sfg_pre) = TRUE & (0d1_1) > 0d1_0 : next(inp_in_start);
      1 : reg_start;
   esac

INIT
sig_ls1 = 0d10_0;
TRANS
next(sig_ls1) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : sig_pk[8:8] ::
            sig_pk[7:7] :: sig_pk[6:6] :: sig_pk[5:5] :: sig_pk[9:9] ::
            sig_pk[3:3] :: sig_pk[2:2] :: sig_pk[1:1] :: sig_pk[0:0] ::
            sig_pk[4:4];
      1 : sig_ls1;
   esac

INIT
sig_ls2 = 0d10_0;
TRANS
next(sig_ls2) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : sig_ls1[7:7] ::
            sig_ls1[6:6] :: sig_ls1[5:5] :: sig_ls1[9:9] ::
            sig_ls1[8:8] :: sig_ls1[2:2] :: sig_ls1[1:1] ::
            sig_ls1[0:0] :: sig_ls1[4:4] :: sig_ls1[3:3];
      1 : sig_ls2;
   esac

INIT
sig_pk = 0d10_0;
TRANS
next(sig_pk) =
   case
      next(sfg_keys) = TRUE & (0d1_1) > 0d1_0 : reg_key[5:5] ::
            reg_key[7:7] :: reg_key[8:8] :: reg_key[0:0] ::
            reg_key[9:9] :: reg_key[3:3] :: reg_key[6:6] ::
            reg_key[1:1] :: reg_key[4:4] :: reg_key[2:2];
      1 : sig_pk;
   esac

INIT
sig_p4 = 0d4_0;
TRANS
next(sig_p4) =
```

```
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : sig_first[0:0] ::
            sig_second[0:0] :: sig_second[1:1] :: sig_first[1:1];
      1 : sig_p4;
   esac

INIT
sig_fkl = 0d4_0;
TRANS
next(sig_fkl) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 :
            reg_left xor next(sig_p4);
      1 : sig_fkl;
   esac

INIT
sig_ep = 0d8_0;
TRANS
next(sig_ep) =
   case
      next(sfg_ip) = TRUE & (0d1_1) > 0d1_0 : reg_right[0:0] ::
            reg_right[3:3] :: reg_right[2:2] :: reg_right[1:1] ::
            reg_right[2:2] :: reg_right[1:1] :: reg_right[0:0] ::
            reg_right[3:3];
      next(sfg_secondf) = TRUE & (0d1_1) > 0d1_0 : reg_right[0:0] ::
            reg_right[3:3] :: reg_right[2:2] :: reg_right[1:1] ::
            reg_right[2:2] :: reg_right[1:1] :: reg_right[0:0] ::
            reg_right[3:3];
      1 : sig_ep;
   esac

INIT
sig_s0r = 0d2_0;
TRANS
next(sig_s0r) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : reg_res[7:7] ::
            reg_res[4:4];
      1 : sig_s0r;
   esac

INIT
sig_s0c = 0d2_0;
TRANS
next(sig_s0c) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : reg_res[6:6] ::
            reg_res[5:5];
      1 : sig_s0c;
   esac

INIT
sig_s1r = 0d2_0;
TRANS
next(sig_s1r) =
   case
```

```
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : reg_res[0:0] ::
            reg_res[3:3];
      1 : sig_s1r;
   esac

INIT
sig_s1c = 0d2_0;
TRANS
next(sig_s1c) =
   case
      next(sfg_slook) = TRUE & (0d1_1) > 0d1_0 : reg_res[2:2] ::
            reg_res[1:1];
      1 : sig_s1c;
   esac

INIT
sig_first = 0d2_0;
TRANS
next(sig_first) =
   case
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_0))) >
            0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_1))) >
            0d1_0 : 0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s0c) = 0d2_2)) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s0c) = 0d2_3)) > 0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_0))) >
            0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_1))) >
            0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s0c) = 0d2_2)) > 0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s0c) = 0d2_3)) > 0d1_0 : 0d1_0 ::
            (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = 0d2_2)) &
            word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_0))) > 0d1_0 :
            0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = 0d2_2)) &
            word1(next(sig_s0c) = (( 0d1_0 ) :: 0d1_1))) > 0d1_0 :
            0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = 0d2_2)) &
            word1(next(sig_s0c) = 0d2_2)) > 0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s0r) = 0d2_2)) &
            word1(next(sig_s0c) = 0d2_3)) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & (word1(next(sig_s0c) = (( 0d1_0 ) ::
            0d1_0))) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & (word1(next(sig_s0c) = (( 0d1_0 ) ::
            0d1_1))) > 0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & (word1(next(sig_s0c) = 0d2_2)) > 0d1_0 :
            0d2_3;
```

```
      next(sfg_slook) = TRUE & (word1(next(sig_s0c) = 0d2_3)) > 0d1_0 :
            0d2_2;
      1 : sig_first;
   esac

INIT
sig_second = 0d2_0;
TRANS
next(sig_second) =
   case
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_0))) >
            0d1_0 : 0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_1))) >
            0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s1c) = 0d2_2)) > 0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_0))) & word1(next(sig_s1c) = 0d2_3)) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_0))) >
            0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_1))) >
            0d1_0 : 0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s1c) = 0d2_2)) > 0d1_0 : 0d1_0 ::
            (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = (( 0d1_0 ) ::
            0d1_1))) & word1(next(sig_s1c) = 0d2_3)) > 0d1_0 : 0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = 0d2_2)) &
            word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_0))) > 0d1_0 :
            0d2_3;
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = 0d2_2)) &
            word1(next(sig_s1c) = (( 0d1_0 ) :: 0d1_1))) > 0d1_0 :
            0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = 0d2_2)) &
            word1(next(sig_s1c) = 0d2_2)) > 0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & ((word1(next(sig_s1r) = 0d2_2)) &
            word1(next(sig_s1c) = 0d2_3)) > 0d1_0 : 0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & (word1(next(sig_s1c) = (( 0d1_0 ) ::
            0d1_0))) > 0d1_0 : 0d2_2;
      next(sfg_slook) = TRUE & (word1(next(sig_s1c) = (( 0d1_0 ) ::
            0d1_1))) > 0d1_0 : 0d1_0 :: (0d1_1);
      next(sfg_slook) = TRUE & (word1(next(sig_s1c) = 0d2_2)) > 0d1_0 :
            0d1_0 :: (0d1_0);
      next(sfg_slook) = TRUE & (word1(next(sig_s1c) = 0d2_3)) > 0d1_0 :
            0d2_3;
      1 : sig_second;
   esac

INIT
sfg_pre = FALSE;
TRANS
next(sfg_pre) =
   case
```

```
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : FALSE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_ini = FALSE;
TRANS
next(sfg_ini) =
   case
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : TRUE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_keys = FALSE;
TRANS
next(sfg_keys) =
   case
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : FALSE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_lr = FALSE;
TRANS
next(sfg_lr) =
   case
```

```
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : FALSE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_ip = FALSE;
TRANS
next(sfg_ip) =
   case
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : FALSE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_slook = FALSE;
TRANS
next(sfg_slook) =
   case
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : FALSE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_rl = FALSE;
TRANS
next(sfg_rl) =
   case
```

```
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : FALSE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_secondf = FALSE;
TRANS
next(sfg_secondf) =
   case
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : FALSE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_setout = FALSE;
TRANS
next(sfg_setout) =
   case
      ctl_decrypt = st_s0 & (reg_start) > 0d1_0 : FALSE;
      ctl_decrypt = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s3 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s4 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s5 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s6 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s7 & (0d1_1) > 0d1_0 : FALSE;
      ctl_decrypt = st_s8 & (0d1_1) > 0d1_0 : TRUE;
      ctl_decrypt = st_s9 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac



MODULE mod_test_encrypt(out_n, out_key)

VAR
```

```
ctl_test_encrypt :  {st_s0};



sfg_always : boolean;

INIT
ctl_test_encrypt = st_s0;
TRANS
next(ctl_test_encrypt) =
   case
      ctl_test_encrypt = st_s0 & (0d1_1) > 0d1_0 : st_s0;
      1 : ctl_test_encrypt;
   esac

INIT
out_n = 0d8_0;
TRANS
next(out_n) =
   case
      next(sfg_always) = TRUE & (0d1_1) > 0d1_0 : 0d1_0 :: (0d7_100);
      1 : out_n;
   esac

INIT
out_key = 0d10_0;
TRANS
next(out_key) =
   case
      next(sfg_always) = TRUE & (0d1_1) > 0d1_0 : 0d8_0 :: (0d2_2);
      1 : out_key;
   esac

INIT
sfg_always = FALSE;
TRANS
next(sfg_always) =
   case
      ctl_test_encrypt = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac
```

A.4 Verification

A.4.1 NuSMV source with CTL specifications defined

```
MODULE main

VAR

sig_m : word[5];
sig_n : word[5];
sig_gcd : word[5];
```

```
um_euclid : mod_euclid(sig_m, sig_n, sig_gcd);
um_test_euclid : mod_test_euclid(sig_m, sig_n);

CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_0)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_11)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_12)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_13)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_14)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_15)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_16)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_17)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_18)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_19)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_20)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_21)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_22)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_23)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_24)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_25)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_26)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_27)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_28)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_29)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_30)
CTLSPEC AG (sig_m = 0d5_0 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_31)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_1 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
```

```
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_2 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_3 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
```

```
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_4 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_5 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
```

```
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_6 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_7 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
```

```
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_8 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_9 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_10 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
```

```
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_11)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_11)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_11)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_11 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_12)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_12)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_12)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_12 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_13)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
```

```
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_13)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_13)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_13 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_14)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_14)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_14)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_14 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_15)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
```

```
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_15)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_15)
CTLSPEC AG (sig_m = 0d5_15 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_16)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_16)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_16 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_17)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_17)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
```

```
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_17 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_18)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_18)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_18 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_19)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_19)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
```

```
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_19 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_20)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_20)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_20 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_21)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_21)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_21 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_22)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
```

```
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_11)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_22)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_22 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_23)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_23)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_23 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_24)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
```

```
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_12)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_8)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_24)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_24 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_25)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_25)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_25 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_26)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_13)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
```

```
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_26)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_26 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_27)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_9)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_27)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_27 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_28)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_14)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_7)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
```

```
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_4)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_28)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_28 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_29)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_29)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_29 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_30)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_15)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_10)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_6)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_5)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_2)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_30)
```

```
CTLSPEC AG (sig_m = 0d5_30 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_0 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_31)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_1 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_2 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_3 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_4 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_5 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_6 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_7 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_8 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_9 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_10 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_11 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_12 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_13 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_14 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_15 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_16 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_17 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_18 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_19 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_20 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_21 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_22 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_23 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_24 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_25 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_26 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_27 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_28 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_29 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_1)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_31)

CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_30 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)
CTLSPEC AG (sig_m = 0d5_31 & sig_n = 0d5_31 -> sig_gcd = 0d5_0 | sig_gcd = 0d5_3)



MODULE mod_euclid(inp_m_in, inp_n_in, out_gcd)

VAR

ctl_euclid :  {st_s0, st_s1, st_s2};

reg_m : word[5];
reg_n : word[5];
reg_done : word[1];
reg_factor : word[5];


sfg_init : boolean;
sfg_shiftm : boolean;
sfg_shiftn : boolean;
sfg_reduce : boolean;
sfg_shiftf : boolean;
sfg_outidle : boolean;
sfg_complete : boolean;

INIT
ctl_euclid = st_s0;
TRANS
next(ctl_euclid) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : st_s2;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : st_s1;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : st_s1;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : st_s2;
```

```
      1 : ctl_euclid;
   esac

INIT
out_gcd = 0d5_0;
TRANS
next(out_gcd) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : 0d4_0 :: (0d1_0);
      next(sfg_outidle) = TRUE & (0d1_1) > 0d1_0 : 0d4_0 :: (0d1_0);
      next(sfg_complete) = TRUE & ((word1(reg_m > reg_n))) > 0d1_0 :
        ((0d3_0 :: ((reg_m))) << ((reg_factor)[2:0]))[4:0];
      next(sfg_complete) = TRUE & (0d1_1) > 0d1_0 : ((0d3_0 :: ((reg_n))) <<
        ((reg_factor)[2:0]))[4:0];
      1 : out_gcd;
   esac

INIT
reg_m = 0d5_0;
TRANS
next(reg_m) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : next(inp_m_in);
      next(sfg_shiftm) = TRUE & (0d1_1) > 0d1_0 : ((0d3_0 :: (reg_m)) >> ((0d1_1)))[4:0];
      next(sfg_reduce) = TRUE & ((word1(reg_m >= reg_n))) > 0d1_0 : reg_m - reg_n;
      next(sfg_reduce) = TRUE & (0d1_1) > 0d1_0 : reg_m;
      1 : reg_m;
   esac

INIT
reg_n = 0d5_0;
TRANS
next(reg_n) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : next(inp_n_in);
      next(sfg_shiftn) = TRUE & (0d1_1) > 0d1_0 : ((0d3_0 :: (reg_n)) >> ((0d1_1)))[4:0];
      next(sfg_reduce) = TRUE & ((word1(reg_n > reg_m))) > 0d1_0 : reg_n - reg_m;
      next(sfg_reduce) = TRUE & (0d1_1) > 0d1_0 : reg_n;
      1 : reg_n;
   esac

INIT
reg_done = 0d1_0;
TRANS
next(reg_done) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : 0d1_0;
      next(sfg_outidle) = TRUE & (0d1_1) > 0d1_0 : ((word1(reg_m = (( 0d4_0 ) :: 0d1_0)))
        | (word1(reg_n = (( 0d4_0 ) :: 0d1_0))));
      1 : reg_done;
   esac

INIT
reg_factor = 0d5_0;
TRANS
next(reg_factor) =
   case
      next(sfg_init) = TRUE & (0d1_1) > 0d1_0 : 0d4_0 :: (0d1_0);
      next(sfg_shiftf) = TRUE & (0d1_1) > 0d1_0 : reg_factor + (( 0d4_0 ) :: 0d1_1);
      1 : reg_factor;
   esac

INIT
sfg_init = FALSE;
TRANS
next(sfg_init) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
```

```
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftm = FALSE;
TRANS
next(sfg_shiftm) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftn = FALSE;
TRANS
next(sfg_shiftn) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_reduce = FALSE;
TRANS
next(sfg_reduce) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_shiftf = FALSE;
TRANS
next(sfg_shiftf) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac

INIT
sfg_outidle = FALSE;
TRANS
next(sfg_outidle) =
   case
```

```
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : TRUE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac

INIT
sfg_complete = FALSE;
TRANS
next(sfg_complete) =
   case
      ctl_euclid = st_s0 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_done) > 0d1_0 : TRUE;
      ctl_euclid = st_s1 & (reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (reg_m[0:0] & !reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (!reg_m[0:0] & reg_n[0:0]) > 0d1_0 : FALSE;
      ctl_euclid = st_s1 & (0d1_1) > 0d1_0 : FALSE;
      ctl_euclid = st_s2 & (0d1_1) > 0d1_0 : FALSE;
      1 : FALSE;
   esac



MODULE mod_test_euclid(out_m, out_n)

VAR

ctl_test_euclid :  {st_s0};



sfg_always : boolean;

INIT
ctl_test_euclid = st_s0;
TRANS
next(ctl_test_euclid) =
   case
      ctl_test_euclid = st_s0 & (0d1_1) > 0d1_0 : st_s0;
      1 : ctl_test_euclid;
   esac


TRANS
next(out_m) =
   case
      next(sfg_always) = TRUE & (0d1_1) > 0d1_0 : out_m;
      1 : out_m;
   esac


TRANS
next(out_n) =
   case
      next(sfg_always) = TRUE & (0d1_1) > 0d1_0 : out_n;
      1 : out_n;
   esac

INIT
sfg_always = FALSE;
TRANS
next(sfg_always) =
   case
      ctl_test_euclid = st_s0 & (0d1_1) > 0d1_0 : TRUE;
      1 : FALSE;
   esac
```

A.4.2 Verification output

```
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_0))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_11))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_12))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_13))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_14))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_15))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_16))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_17))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_18))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_19))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_20))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_21))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_22))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_23))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_24))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_25))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_26))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_27))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_28))  is true
```

```
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_29))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_30))  is true
-- specification AG ((sig_m = 0d5_0 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_31))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_1 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
```

```
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_2 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_3 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
```

```
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_4 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_5 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
```

```
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_6 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
```

```
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_7 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
```

```
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_8 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_9 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
```

```
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_10 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_11))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_11))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_11))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_11 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_12))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_12))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
```

```
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_12))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_12 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_13))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_13))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_13))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_13 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_14))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_14))  is true
```

```
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_14))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_14 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_15))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_15))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_15))  is true
-- specification AG ((sig_m = 0d5_15 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_16))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
```

```
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_16))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_16 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_17))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_17))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_17 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_18))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
```

```
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_18))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_18 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_19))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_19))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_19 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_20))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
```

```
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_20))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_20 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_21))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
```

```
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_21))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_21 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_22))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
```

```
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_11))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_22))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_22 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_23))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_23))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_23 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_24))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
```

```
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_12))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_8))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_24))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_24 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_25))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_25))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_25 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_26))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
```

```
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_13))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_26))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_26 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_27))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_9))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_27))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_27 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_28))  is true
```

```
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_14))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_7))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_4))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_28))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_28 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_29))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_29))  is true
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_29 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_30))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_15))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_10))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_6))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_5))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_2))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_30))  is true
-- specification AG ((sig_m = 0d5_30 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_0) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_31))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_1) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_2) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_3) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_4) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_5) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_6) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_7) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_8) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_9) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_10) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_11) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_12) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_13) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_14) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_15) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_16) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_17) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_18) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_19) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_20) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_21) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_22) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_23) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_24) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_25) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_26) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_27) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_28) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
```

```
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_29) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_1))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_31))  is true
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_30) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  sig_m = 0d5_31
  sig_n = 0d5_30
  sig_gcd = 0d5_0
  um_euclid.ctl_euclid = st_s0
  um_euclid.reg_m = 0d5_0
  um_euclid.reg_n = 0d5_0
  um_euclid.reg_done = 0d1_0
  um_euclid.reg_factor = 0d5_0
  um_euclid.sfg_init = 0
  um_euclid.sfg_shiftm = 0
  um_euclid.sfg_shiftn = 0
  um_euclid.sfg_reduce = 0
  um_euclid.sfg_shiftf = 0
  um_euclid.sfg_outidle = 0
  um_euclid.sfg_complete = 0
  um_test_euclid.sfg_always = 0
  um_test_euclid.ctl_test_euclid = st_s0
-> Input: 1.2 <-
-> State: 1.2 <-
  um_euclid.ctl_euclid = st_s1
  um_euclid.reg_m = 0d5_31
  um_euclid.reg_n = 0d5_30
  um_euclid.sfg_init = 1
  um_test_euclid.sfg_always = 1
-> Input: 1.3 <-
-> State: 1.3 <-
  um_euclid.reg_n = 0d5_15
  um_euclid.sfg_init = 0
  um_euclid.sfg_shiftn = 1
  um_euclid.sfg_outidle = 1
-> Input: 1.4 <-
-> State: 1.4 <-
  um_euclid.reg_m = 0d5_16
  um_euclid.sfg_shiftn = 0
  um_euclid.sfg_reduce = 1
-> Input: 1.5 <-
-> State: 1.5 <-
  um_euclid.reg_m = 0d5_8
  um_euclid.sfg_shiftm = 1
  um_euclid.sfg_reduce = 0
-> Input: 1.6 <-
-> State: 1.6 <-
  um_euclid.reg_m = 0d5_4
-> Input: 1.7 <-
-> State: 1.7 <-
  um_euclid.reg_m = 0d5_2
-> Input: 1.8 <-
-> State: 1.8 <-
  um_euclid.reg_m = 0d5_1
-> Input: 1.9 <-
-> State: 1.9 <-
```

```
  um_euclid.reg_n = 0d5_14
  um_euclid.sfg_shiftm = 0
  um_euclid.sfg_reduce = 1
-> Input: 1.10 <-
-> State: 1.10 <-
  um_euclid.reg_n = 0d5_7
  um_euclid.sfg_shiftn = 1
  um_euclid.sfg_reduce = 0
-> Input: 1.11 <-
-> State: 1.11 <-
  um_euclid.reg_n = 0d5_6
  um_euclid.sfg_shiftn = 0
  um_euclid.sfg_reduce = 1
-> Input: 1.12 <-
-> State: 1.12 <-
  um_euclid.reg_n = 0d5_3
  um_euclid.sfg_shiftn = 1
  um_euclid.sfg_reduce = 0
-> Input: 1.13 <-
-> State: 1.13 <-
  um_euclid.reg_n = 0d5_2
  um_euclid.sfg_shiftn = 0
  um_euclid.sfg_reduce = 1
-> Input: 1.14 <-
-> State: 1.14 <-
  um_euclid.reg_n = 0d5_1
  um_euclid.sfg_shiftn = 1
  um_euclid.sfg_reduce = 0
-> Input: 1.15 <-
-> State: 1.15 <-
  um_euclid.reg_m = 0d5_0
  um_euclid.sfg_shiftn = 0
  um_euclid.sfg_reduce = 1
-> Input: 1.16 <-
-> State: 1.16 <-
  um_euclid.reg_done = 0d1_1
  um_euclid.sfg_shiftm = 1
  um_euclid.sfg_reduce = 0
-> Input: 1.17 <-
-> State: 1.17 <-
  sig_gcd = 0d5_1
  um_euclid.ctl_euclid = st_s2
  um_euclid.sfg_shiftm = 0
  um_euclid.sfg_outidle = 0
  um_euclid.sfg_complete = 1
-- specification AG ((sig_m = 0d5_31 & sig_n = 0d5_31) -> (sig_gcd = 0d5_0 |
sig_gcd = 0d5_3))  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 2.1 <-
  sig_m = 0d5_31
  sig_n = 0d5_31
  sig_gcd = 0d5_0
  um_euclid.ctl_euclid = st_s0
  um_euclid.reg_m = 0d5_0
  um_euclid.reg_n = 0d5_0
  um_euclid.reg_done = 0d1_0
  um_euclid.reg_factor = 0d5_0
  um_euclid.sfg_init = 0
  um_euclid.sfg_shiftm = 0
  um_euclid.sfg_shiftn = 0
  um_euclid.sfg_reduce = 0
```

```
  um_euclid.sfg_shiftf = 0
  um_euclid.sfg_outidle = 0
  um_euclid.sfg_complete = 0
  um_test_euclid.sfg_always = 0
  um_test_euclid.ctl_test_euclid = st_s0
-> Input: 2.2 <-
-> State: 2.2 <-
  um_euclid.ctl_euclid = st_s1
  um_euclid.reg_m = 0d5_31
  um_euclid.reg_n = 0d5_31
  um_euclid.sfg_init = 1
  um_test_euclid.sfg_always = 1
-> Input: 2.3 <-
-> State: 2.3 <-
  um_euclid.reg_m = 0d5_0
  um_euclid.sfg_init = 0
  um_euclid.sfg_reduce = 1
  um_euclid.sfg_outidle = 1
-> Input: 2.4 <-
-> State: 2.4 <-
  um_euclid.reg_done = 0d1_1
  um_euclid.sfg_shiftm = 1
  um_euclid.sfg_reduce = 0
-> Input: 2.5 <-
-> State: 2.5 <-
  sig_gcd = 0d5_31
  um_euclid.ctl_euclid = st_s2
  um_euclid.sfg_shiftm = 0
  um_euclid.sfg_outidle = 0
  um_euclid.sfg_complete = 1
```

A graphical analysis tool for hardware design

Appendix B

Mariusz Pawel Pytel
s030166

Kongens Lyngby 2007
IMM-M.Sc-2007-111

Technical University of Denmark
Informatics and Mathematical Modeling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-MSc:2007-111

Table of Contents

Appendix B

B.1 Package maaoha

B.1.1 MyTool.java

```
package maaoha;

import javax.swing.JApplet;
import javax.swing.JFrame;

import maaoha.view.ToolView;

/*
 * Class responsible for creation of main window and instantiation
 *  of the tool
 */
public class MyTool extends JApplet {
  public static JFrame frame;

  static final long serialVersionUID = 1;

  /*
   * Application's main function. Create
   */
  public static void main(String[] args) {
    frame = new JFrame("MaAoHA");
    ToolView myTool = new ToolView();
    frame.getContentPane().add(myTool);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
    frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
  }
}
```

B.1.2 MaaohaException.java

```
package maaoha;

/*
 * Exception class for Maaoha tool
 */
public class MaaohaException extends Exception
{
  static final long serialVersionUID = 1;

  public MaaohaException(String msg)
  {
    super(msg);
  }
}
```

B.2 Package maaoha.model

B.2.1 Condition.java

```
package maaoha.model;

import java.util.ArrayList;

import maaoha.MaaohaException;
import maaoha.model.expression.BasicExpression;
import maaoha.tools.MaaohaTools;

/*
 * Class responsible for representaton of condition in the model
 * structure
 */
public class Condition {
  protected BasicExpression expression;

  /*
   * Creates a new instance of the class with specified value and elements
   * that may appear in the expression
   */
  public Condition(String value, ArrayList<DatapathElement> elements,
      ArrayList<LookUpTable> lookupTables) throws MaaohaException {
    value = MaaohaTools.separateTokens(value);
    this.expression = BasicExpression.createBasicExpression(value,
        elements, lookupTables);
  }

  /*
   * Returns expression value
   */
  public BasicExpression getBasicExpression() {
    return expression;
  }

  /*
   * Sets new expression value
   */
  public void setConditionFromString(String expression,
      ArrayList<DatapathElement> elements,
      ArrayList<LookUpTable> lookupTables) throws MaaohaException {
    expression = MaaohaTools.separateTokens(expression);
    this.expression = BasicExpression.createBasicExpression(expression,
        elements, lookupTables);
  }
}
```

B.2.2 Datapath.java

```
package maaoha.model;

import java.util.ArrayList;
import java.util.StringTokenizer;

import maaoha.MaaohaException;
import maaoha.tools.MaaohaTools;

/*
 * Class responsible for representation of datapath in the model structure
 */
public class Datapath {
  protected ArrayList<LookUpTable> lookupTables;

  protected ArrayList<DatapathElement> elements;
```

```
  protected ArrayList<Sfg> sfgs;

  protected ArrayList<UseModule> useModules;

  protected Module parentModule;

  /*
   * Creates a new instance of the class
   */
  public Datapath(Module parentModule) {
    this.parentModule = parentModule;
    elements = new ArrayList<DatapathElement>();
    sfgs = new ArrayList<Sfg>();
    useModules = new ArrayList<UseModule>();
    lookupTables = new ArrayList<LookUpTable>();
  }

  /*
   * Adds element to datapath
   */
  public void addDatapathElement(DatapathElement de) {
    elements.add(de);
  }

  /*
   * Parses string and adds expression to the specified sfg
   */
  public ArrayList<Expression> addExpressionFromString(Sfg sfg,
      String expressionsToParse) throws MaaohaException {
    ArrayList<Expression> expressionList =
createExpressionListFromString(expressionsToParse);
    for (int i = 0; i < expressionList.size(); i++) {
      sfg.expressions.add(expressionList.get(i));
    }

    return expressionList;
  }

  /*
   * Adds new lookup table
   */
  public void addNewLookUpTable(String name, String type,
      ArrayList<Integer> elements) {
    lookupTables.add(new LookUpTable(name, type, elements));
  }

  /*
   * Adds a signal flow graph to datapath
   */
  public void addSfg(Sfg sfg) {
    sfgs.add(sfg);
  }

  /*
   * Parses string and adds signal flow graph
   */
  public Sfg addSfgFromString(String name, String expressionsToParse)
      throws MaaohaException {
    ArrayList<Expression> expressionList =
createExpressionListFromString(expressionsToParse);
    Sfg sfg = new Sfg(name, expressionList);
    sfgs.add(sfg);
    return sfg;
  }

  /*
   * Adds sub-module
```

```
   */
  public UseModule addUseModule(String name, Module module,
      String parameterList) throws MaaohaException {
    ArrayList<DatapathElement> parameters =
parseUseModuleParameterList(parameterList);
    UseModule um = new UseModule(name, module, parameters);
    useModules.add(um);
    return um;
  }

  /*
   * Adds sub-module to datapath
   */
  public void addUseModule(UseModule useModule) {
    useModules.add(useModule);
  }

  /*
   * Parses string and creates list of expressions
   */
  public ArrayList<Expression> createExpressionListFromString(
      String expressionsToParse) throws MaaohaException {
    ArrayList<Expression> expressionList = new ArrayList<Expression>();

    expressionsToParse = MaaohaTools.separateTokens(expressionsToParse);

    StringTokenizer tokenizer = new StringTokenizer(expressionsToParse);
    String token, strExpression;
    while (tokenizer.hasMoreTokens()) {
      strExpression = "";
      boolean leftSide = true;
      DatapathElement leftSideElement = null;
      ArrayList<DatapathElement> rightSideElements = new
ArrayList<DatapathElement>();

      token = tokenizer.nextToken();
      while (!token.equalsIgnoreCase(";") && tokenizer.hasMoreTokens()) {
        if (token.equalsIgnoreCase("="))
          leftSide = false;

        for (int i = 0; i < elements.size(); i++) {
          if (elements.get(i).name.equalsIgnoreCase(token)) {
            if (leftSide)
              leftSideElement = elements.get(i);
            else {
              DatapathElement de = elements.get(i);
              if (!rightSideElements.contains(de))
                rightSideElements.add(de);
            }

            i = elements.size();
          }
        }

        strExpression += token;

        token = tokenizer.nextToken();
      }

      if (leftSideElement == null)
        throw new MaaohaException(
            "Module->createExpressionListFromString::
             Left side of equation has to be an output, signal or register");

      Expression exp = new Expression(strExpression, leftSideElement,
          rightSideElements, getLookupTables());

      expressionList.add(exp);
```

```
    }

    return expressionList;
  }

  /*
   * Checks if element with given name exists
   */
  public boolean elementExists(String name) {
    for (int i = 0; i < elements.size(); i++) {
      if (elements.get(i).name.equals(name))
        return true;
    }

    for (int i = 0; i < sfgs.size(); i++) {
      if (sfgs.get(i).name.equals(name))
        return true;
    }

    for (int i = 0; i < useModules.size(); i++) {
      if (useModules.get(i).name.equals(name))
        return true;
    }

    return false;
  }

  /*
   * Returns i-th element of datapath
   */
  public DatapathElement getElement(int i) {
    return elements.get(i);
  }

  /*
   * Return index of the specified element
   */
  public int getElementIndex(DatapathElement de) {
    return elements.indexOf(de);
  }

  /*
   * Returns list of datapath elements
   */
  public ArrayList<DatapathElement> getElements() {
    return elements;
  }

  /*
   * Returns number of datapath elements
   */
  public int getElementsNo() {
    return elements.size();
  }

  /*
   * Returns index of the specified input or output
   */
  public int getInputOutputIndex(DatapathElement de) {
    int index = 0;
    if (!de.getElementType().equals(Model.typeOutput)
        && !de.getElementType().equals(Model.typeInput))
      return -1;
    else {
      for (int i = 0; i < elements.size(); i++) {
        if (elements.get(i).getElementType().equals(Model.typeOutput)
            || elements.get(i).getElementType().equals(
                Model.typeInput)) {
```

```
        if (de == elements.get(i))
          return index;
        else
          index++;
      }
    }
  }
  return -1;
}

/*
 * Returns type of i-th input output
 */
public String getInputOutputType(int index) {
  for (int i = 0; i < elements.size(); i++) {
    if (elements.get(i).elementType.equals(Model.typeInput)
        || elements.get(i).elementType.equals(Model.typeOutput)) {
      if (index == 0)
        return elements.get(i).elementType;
      else
        index--;
    }
  }
  return null;
}

/*
 * Returns number of inputs in datpath
 */
public int getInputsNo() {
  int result = 0;
  for (int i = 0; i < elements.size(); i++)
    if (elements.get(i).elementType.equals(Model.typeInput))
      result++;
  return result;
}

/*
 * Returns list of lookup tables
 */
public ArrayList<LookUpTable> getLookupTables() {
  return lookupTables;
}

/*
 * Returns i-th output
 */
public DatapathElement getOutputByIndex(int index) {
  for (int i = 0; i < elements.size(); i++) {
    if (elements.get(i).getElementType().equals(Model.typeOutput)) {
      if (index-- == 0)
        return elements.get(i);
    }
  }
  return null;
}

/*
 * Returns index of the specified output
 */
public int getOutputIndex(DatapathElement de) {
  int index = 0;
  if (!de.getElementType().equals(Model.typeOutput))
    return -1;
  else {
    for (int i = 0; i < elements.size(); i++) {
      if (elements.get(i).getElementType().equals(Model.typeOutput)) {
        if (de == elements.get(i))
```

```
          return index;
        else
          index++;
      }
    }
  }
  return -1;
}

/*
 * Return number of outputs in datapath
 */
public int getOutputsNo() {
  int result = 0;
  for (int i = 0; i < elements.size(); i++)
    if (elements.get(i).elementType.equals(Model.typeOutput))
      result++;
  return result;
}

/*
 * Returns i-th signal flow graph
 */
public Sfg getSfg(int i) {
  return sfgs.get(i);
}

/*
 * Returns number of signal flow graphs
 */
public int getSfgsNo() {
  return sfgs.size();
}

/*
 * Returns total sum of expressions for all signal flow graphs
 */
public int getTotalExpressionNo() {
  int sum = 0;

  for (int i = 0; i < sfgs.size(); i++)
    sum += sfgs.get(i).getExpressionsNo();

  return sum;
}

/*
 * Returns i-th sub-module
 */
public UseModule getUseModule(int i) {
  return useModules.get(i);
}

/*
 * Returns number of sub-modules used in datapath
 */
public int getUseModulesNo() {
  return useModules.size();
}

/*
 * Checks whether datapath has input or output
 */
public boolean hasInputsOutputs() {
  for (int i = 0; i < elements.size(); i++)
    if (elements.get(i).elementType.equals(Model.typeInput)
        || elements.get(i).elementType.equals(Model.typeOutput))
      return true;
```

```
  return false;
}

/*
 * Verifies wheteher an element is input or output for sub-module
 */
public boolean isElementUsedAsOutputForModule(DatapathElement de) {
  for (int i = 0; i < getUseModulesNo(); i++) {
    for (int j = 0; j < getUseModule(i).getParametersNo(); j++) {
      if ((getUseModule(i).getParameter(j) == de)
          && getUseModule(i).getModule().getDatapath()
              .isParameterOutput(j))
        return true;
    }
  }
  return false;
}

/*
 * Verifies whether element of the given idex is an output
 */
public boolean isParameterOutput(int index) {
  for (int i = 0; i < elements.size(); i++) {
    if (elements.get(i).getElementType().equals(Model.typeOutput)
        || elements.get(i).getElementType().equals(Model.typeInput)) {
      if (index == 0) {
        if (elements.get(i).getElementType().equals(
            Model.typeOutput))
          return true;
        else
          return false;
      } else
        index--;
    }
  }
  return false;
}

/*
 * Parses parameter list for sub-module
 */
public ArrayList<DatapathElement> parseUseModuleParameterList(
    String parameterList) {
  ArrayList<DatapathElement> parameters = new ArrayList<DatapathElement>();
  parameterList = parameterList.replaceAll(",", " , ");
  StringTokenizer tokenizer = new StringTokenizer(parameterList);
  String token;
  while (tokenizer.hasMoreTokens()) {
    token = tokenizer.nextToken();
    if (!token.equalsIgnoreCase(",")) {
      for (int i = 0; i < elements.size(); i++) {
        if (elements.get(i).name.equalsIgnoreCase(token)) {
          parameters.add(elements.get(i));
          i = elements.size();
        }
      }
    }
  }
  return parameters;
}

/*
 * Removes an element from datapath. Returns list of use modules that are
 * using given element as a parameter
 */
public ArrayList<UseModuleAndParameter> removeElement(
    DatapathElement element) {
```

```
    // looks for sub-modules that use this element
    for (int i = 0; i < useModules.size(); i++) {
      UseModule um = useModules.get(i);
      for (int j = 0; j < um.getParametersNo(); j++) {
        if (um.getParameter(j) == element)
          um.setParameter(j, null);

      }
    }
    // looks for expressions that use this element
    for (int i = 0; i < sfgs.size(); i++) {
      Sfg sfg = sfgs.get(i);
      for (int j = 0; j < sfg.getExpressionsNo(); j++) {
        Expression exp = sfg.getExpression(j);
        if (exp.getLeftSideElement() == element)
          exp.setLeftSideElement(null);
        for (int k = 0; k < exp.getRightSideElementsNo(); k++)
          if (exp.getRightSideElement(k) == element)
            exp.setRightSideElement(k, null);
      }
    }
    // if element is an input or output
    // list of module instantions have to be updated
    ArrayList<UseModuleAndParameter> formerParameters = null;
    if (element.elementType.equals(Model.typeOutput)
        || element.elementType.equals(Model.typeInput)) {
      formerParameters = parentModule.getParentModel()
          .removeElementFromUseModules(parentModule,
              getInputOutputIndex(element));
    }
    elements.remove(element);
    return formerParameters;
  }

  /*
   * Removes sfg
   */
  public void removeSfg(Sfg sfg) {
    sfgs.remove(sfg);
  }

  /*
   * Removes sub-module
   */
  public void removeUseModule(UseModule useModule) {
    useModules.remove(useModule);
  }

  /*
   * Sets expression list for specified sfg
   */
  public void setSfgExpressions(Sfg sfg, ArrayList<Expression> expressionList) {
    sfg.setExpressions(expressionList);
  }
}
```

B.2.3 DatapathElement.java

```
package maaoha.model;

/*
 * Class responsible for representation of datapah element in
 * the model structure
 */
public class DatapathElement {
      protected String name;
```

```
	protected String elementType;

	protected String valueType;

	/*
	 * Creates a new instance of datapath element
	 */
	public DatapathElement(String name, String elementType, String valueType) {
		this.name = name;
		this.elementType = elementType;
		this.valueType = valueType;
	}

	/*
	 * Returns element type
	 */
	public String getElementType() {
		return elementType;
	}

	/*
	 * Returns element name
	 */
	public String getName() {
		return name;
	}

	/*
	 * Returns element value type
	 */
	public String getValueType() {
		return valueType;
	}

	/*
	 * Sets element name
	 */
	public void setName(String name) {
		this.name = name;
	}

	/*
	 * Sets element value type
	 */
	public void setValueType(String valueType) {
		this.valueType = valueType;
	}
}
```

B.2.4 Expression.java

```
package maaoha.model;

import java.util.ArrayList;

import maaoha.MaaohaException;
import maaoha.model.expression.BasicExpression;
import maaoha.tools.MaaohaTools;

/*
 * Class responsible for representation of the expression in the model
 * structure
 */
public class Expression {
  protected DatapathElement leftSideElement;
```

```
protected ArrayList<DatapathElement> rightSideElements;

protected int id;

protected BasicExpression expression;

/*
 * Creates a new instance of expression
 */
public Expression(String value, DatapathElement leftSideElement,
    ArrayList<DatapathElement> rightSideElements,
    ArrayList<LookUpTable> lookupTables) throws MaaohaException {
  this.leftSideElement = leftSideElement;
  this.rightSideElements = rightSideElements;
  value = MaaohaTools.separateTokens(value);
  value = MaaohaTools.cutLeftSide(value);
  this.expression = BasicExpression.createBasicExpression(value,
      rightSideElements, lookupTables);
}

/*
 * Checks whether all elements all defined
 */
public boolean allElementsDefined() {
  if (leftSideElement == null)
    return false;
  for (int i = 0; i < rightSideElements.size(); i++)
    if (rightSideElements.get(i) == null)
      return false;

  return true;
}

/*
 * Returns expression value
 */
public BasicExpression getBasicExpression() {
  return expression;
}

/*
 * Returns expression left side element
 */
public DatapathElement getLeftSideElement() {
  return leftSideElement;
}

/*
 * Returns list of elements that appear on the right side of expression
 */
public DatapathElement getRightSideElement(int i) {
  return rightSideElements.get(i);
}

/*
 * Returns number of elements that appear on the right side of expression
 */
public int getRightSideElementsNo() {
  return rightSideElements.size();
}

/*
 * Returns expression value
 */
public String getValue() {
  if (leftSideElement == null)
    return "???=" + expression.toString();
  else
```

```
      return leftSideElement.getName() + "=" + expression.toString();
    }

    /*
     * Sets element on the left side of expression
     */
    public void setLeftSideElement(DatapathElement de) {
      leftSideElement = de;
    }

    /*
     * Sets i-th element on the right side of expression
     */
    public void setRightSideElement(int i, DatapathElement de) {
      expression.setElement(rightSideElements.remove(i), de);
      rightSideElements.add(i, de);
    }
}
```

B.2.5 Fsm.java

```
package maaoha.model;

import java.util.ArrayList;

/*
 * Class responsible for representation of controller in the model
 * structure
 */
public class Fsm {
    protected FsmState initialState;

    protected ArrayList<FsmState> states;

    /*
     * Creates a new instance of controller
     */
    public Fsm() {
      states = new ArrayList<FsmState>();
    }

    /*
     * Adds a state to the controller
     */
    public void addFsmState(FsmState state) {
      states.add(state);
    }

    /*
     * Adds a transition to a state
     */
    public FsmTransition addTransitionToState(FsmState state,
        Condition condition, ArrayList<Sfg> usedSfgs, FsmState nextState) {
      return state.addTransition(condition, usedSfgs, nextState);
    }

    /*
     * Creates a new state and adds it to the controller
     */
    public FsmState createNewState(String name) {
      FsmState state = new FsmState(name);
      states.add(state);
      return state;
    }

    /*
     * Returns i-th state
```

```
   */
  public FsmState getFsmState(int i) {
    return states.get(i);
  }

  /*
   * Returns the initial state
   */
  public FsmState getInitialState() {
    return initialState;
  }

  /*
   * Returns number of controller states
   */
  public int getStatesNo() {
    return states.size();
  }

  /*
   * Returns total number of controller transitons
   */
  public int getTotalTransitionsNo() {
    int sum = 0;
    for (int i = 0; i < states.size(); i++) {
      sum += states.get(i).getTransitionsNo();
    }
    return sum;
  }

  /*
   * Removes state from controller
   */
  public void removeFsmState(FsmState fsmState) {
    states.remove(fsmState);
  }

  /*
   * Sets the initial state
   */
  public void setInitialState(FsmState initialState) {
    this.initialState = initialState;
  }

  /*
   * Checks whether state of given name exists
   */
  public boolean stateExist(String name) {
    for (int i = 0; i < states.size(); i++) {
      if (states.get(i).name.equals(name))
        return true;
    }
    return false;
  }
}
```

B.2.6 FsmState.java

```
package maaoha.model;

import java.util.ArrayList;

/*
 * Class responsible for representation of the controller state
 * in the model structure
 */
public class FsmState {
```

```
protected String name;

protected ArrayList<FsmTransition> transitions;

/*
 * Creates a new instance of the controller state
 */
public FsmState(String name) {
  this.name = name;
  transitions = new ArrayList<FsmTransition>();
}

/*
 * Adds new transition to the state
 */
public FsmTransition addTransition(Condition condition,
    ArrayList<Sfg> usedSfgs, FsmState nextState) {
  FsmTransition trans = new FsmTransition(condition, usedSfgs, nextState);
  transitions.add(trans);
  return trans;
}

/*
 * Adds new transition to the state
 */
public void addTransition(FsmTransition trans) {
  transitions.add(trans);
}

/*
 * Returns name of the state
 */
public String getName() {
  return name;
}

/*
 * Returns number of transitions to the specified state with possible
 * limitation on transition index
 */
public int getNumberOfTransitionToDestination(FsmState destination,
    int limit) {
  int number = 0;
  if (limit < 0 || limit > transitions.size()) {
    for (int i = 0; i < transitions.size(); i++)
      if (transitions.get(i).nextState == destination)
        number++;
  } else {
    for (int i = 0; i < limit; i++)
      if (transitions.get(i).nextState == destination)
        number++;
  }
  return number;
}

/*
 * Returns i-th transition
 */
public FsmTransition getTransistion(int i) {
  return transitions.get(i);
}

/*
 * Returns transition index
 */
public int getTransitionIndex(FsmTransition trans) {
  return transitions.indexOf(trans);
}
```

```
  /*
   * Returns number of transitions
   */
  public int getTransitionsNo() {
    return transitions.size();
  }

  /*
   * Removes sfg from transitions
   */
  public void removeSfg(Sfg sfg) {
    for (int i = 0; i < transitions.size(); i++) {
      transitions.get(i).removeSfg(sfg);
    }
  }

  /*
   * Removes transition
   */
  public void removeTransition(FsmTransition fsmTransition) {
    transitions.remove(fsmTransition);
  }
}
```

B.2.7 FsmTransition.java

```
package maaoha.model;

import java.util.ArrayList;

/*
 * Class responsible for representation of controller transition
 * in the model structure
 */
public class FsmTransition {
  protected Condition condition;

  protected ArrayList<Sfg> usedSfgs;

  protected FsmState nextState;

  /*
   * Creates new instance of the class
   */
  public FsmTransition(Condition condition, ArrayList<Sfg> usedSfgs,
      FsmState nextState) {
    this.nextState = nextState;
    this.condition = condition;
    this.usedSfgs = usedSfgs;
  }

  /*
   * Checks whether given sfg is executed during the transition
   */
  public boolean containsSfg(Sfg sfg) {
    return usedSfgs.contains(sfg);
  }

  /*
   * Returns transition condition
   */
  public Condition getCondition() {
    return condition;
  }

  /*
```

```
 * Returns number of expressions being executed during transition
 */
public int getExpressionsNo() {
  int sum = 0;
  for (int i = 0; i < usedSfgs.size(); i++) {
    sum += usedSfgs.get(i).getExpressionsNo();
  }
  return sum;
}

/*
 * Returns transition label
 */
public String getName() {
  String name = getSfgsString() + " -> " + nextState.name;
  return name;
}

/*
 * Returns next state for the transition
 */
public FsmState getNextState() {
  return nextState;
}

/*
 * Returns i-th sfg
 */
public Sfg getSfg(int i) {
  return usedSfgs.get(i);
}

/*
 * Return executed sfgs
 */
public ArrayList<Sfg> getSfgs() {
  return usedSfgs;
}

/*
 * Returns names of executed sfgs in string
 */
public String getSfgsString() {
  String res = "";
  if (usedSfgs.size() > 0 && usedSfgs.get(0) != null)
    res += usedSfgs.get(0).name;
  for (int i = 1; i < usedSfgs.size(); i++)
    res += "," + usedSfgs.get(i).name;

  return res;
}

/*
 * Returns names of executed sfgs in string
 */
public String getSfgsString_() {
  String res = "";
  if (usedSfgs.get(0) != null)
    res += usedSfgs.get(0).name;
  for (int i = 1; i < usedSfgs.size(); i++)
    res += "_" + usedSfgs.get(i).name;

  return res;
}

/*
 * Removes sfg from the list of executed sfgs
 */
```

```
  public void removeSfg(Sfg sfg) {
    usedSfgs.remove(sfg);
  }

  /*
   * Sets next state
   */
  public void setNextState(FsmState nextState) {
    this.nextState = nextState;
  }

  /*
   * Sets list of executed sfgs
   */
  public void setSfgs(ArrayList<Sfg> usedSfgs) {
    this.usedSfgs = usedSfgs;
  }

  /*
   * Returns number of executed sfg
   */
  public int usedSfgsNo() {
    return usedSfgs.size();
  }
}
```

B.2.8 LookUpTable.java

```
package maaoha.model;

import java.util.ArrayList;

/*
 * Class responsible for representation of lookup table
 * in the model structure
 */
public class LookUpTable {
  protected String name;

  protected String type;

  protected ArrayList<Integer> elements;

  /*
   * Creates an instance of the class
   */
  public LookUpTable(String name, String type) {
    this.name = name;
    this.type = type;
    elements = new ArrayList<Integer>();
  }

  /*
   * Creates an instance of the class
   */
  public LookUpTable(String name, String type, ArrayList<Integer> elements) {
    this.name = name;
    this.type = type;
    this.elements = elements;
  }

  /*
   * Returns i-th element
   */
  public int getElement(int i) {
    return elements.get(i);
  }
```

```
  /*
   * Returns number of elements
   */
  public int getElementsNo() {
    return elements.size();
  }

  /*
   * Returns name
   */
  public String getName() {
    return name;
  }

  /*
   * Returns type
   */
  public String getType() {
    return type;
  }
}
```

B.2.9 Model.java

```
package maaoha.model;

import java.util.ArrayList;

import maaoha.MaaohaException;

/*
 * Class encapsulates the whole model structure
 */
public class Model {
  public static final String typeRegister = "register";
  public static final String typeSignal = "signal";
  public static final String typeInput = "input";
  public static final String typeOutput = "output";
  public static final String typeModule = "module";
  public static final String typeExpression = "expression";
  public static final String typeState = "state";
  public static final String typeTransition = "transition";

  protected ArrayList<Module> modules;

  protected Module systemModule;

  protected int ExpressionCounter = 0;

  /*
   * Creates new instance of the class
   */
  public Model() throws MaaohaException {
    modules = new ArrayList<Module>();
  }

  /*
   * Associates controller with module
   */
  public void addFsmToModule(Fsm fsm, String moduleName) {
    getModule(moduleName).setFsm(fsm);
  }

  /*
   * Adds module to the system
   */
```

```
public void addModule(Module module) {
  modules.add(module);
}

/*
 * Adds module to the system and sets it as tha main one
 */
public void addSystemModule(Module module) {
  modules.add(0, module);
}

/*
 * Returns main module
 */
public Module getFirstModule() {
  if (modules.get(0) != null)
    return modules.get(0);

  return null;
}

/*
 * Returns i-th module
 */
public Module getModule(int index) {
  return modules.get(index);
}

/*
 * Returns module by name
 */
public Module getModule(String name) {
  for (int i = 0; i < modules.size(); i++)
    if (modules.get(i).name.equals(name))
      return modules.get(i);

  return null;
}

/*
 * Returns number of modules
 */
public int getModulesNo() {
  return modules.size();
}

/*
 * Removes elements that are no longer paramaeters
 */
public ArrayList<UseModuleAndParameter> removeElementFromUseModules(
    Module module, int index) {
  // list of elements that are no longer parameters
  ArrayList<UseModuleAndParameter> elements =
    new ArrayList<UseModuleAndParameter>();
  for (int i = 0; i < modules.size(); i++) {
    Datapath dp = modules.get(i).getDatapath();
    for (int j = 0; j < dp.getUseModulesNo(); j++) {
      if (dp.getUseModule(j).getModule() == module) {
        UseModule um = dp.getUseModule(j);
        elements.add(new UseModuleAndParameter(um
            .getParameter(index), um, modules.get(i)));
        um.removeParameter(index);
      }
    }
  }
  return elements;
}
```

```
/*
 * Removes module
 */
public void removeModule(Module module) {
  modules.remove(module);
}

/*
 * Sets module as of a given name as a system module
 */
public void setAsSystemModule(String name) {
  Module mod = getModule(name);
  systemModule = mod;
  modules.remove(mod);
  modules.add(0, mod);
}

public String toString() {
  String str = "";

  for (int i = 0; i < modules.size(); i++) {
    Module module = modules.get(i);
    str += "MODULE " + module.name + "\n";
    str += "Datapatath\n";
    str += "Sfgs:\n";
    for (int j = 0; j < module.datapath.sfgs.size(); j++) {
      str += "  " + module.datapath.sfgs.get(j).name + "\n";
      for (int k = 0; k < module.datapath.sfgs.get(j).expressions
          .size(); k++) {
        str += "    "
            + module.datapath.sfgs.get(j).expressions.get(k)
                .getValue();
        if (module.datapath.sfgs.get(j).expressions.get(k).leftSideElement
            != null)
          str += " : "
              + module.datapath.sfgs.get(j).expressions
                  .get(k).leftSideElement.name;
        else
          str += "???";
        str += " -> ";
        for (int l = 0; l < module.datapath.sfgs.get(j).expressions
            .get(k).rightSideElements.size(); l++)
          if (module.datapath.sfgs.get(j).expressions.get(k).rightSideElements
              .get(l) != null)
            str += " "
                + module.datapath.sfgs.get(j).expressions
                    .get(k).rightSideElements.get(l).name;
          else
            str += "???";

        str += "\n";
      }

    }
    str += "\n";
    str += "Elements:\n";
    for (int j = 0; j < module.datapath.elements.size(); j++) {
      str += "  " + module.datapath.elements.get(j).name + "\n";
    }
    str += "\n";

    str += "Use Module:\n";
    for (int j = 0; j < module.datapath.useModules.size(); j++) {
      str += "  "
          + module.datapath.useModules.get(j).getModule().name
          + "\n";
    }
    str += "\n";
```

```
      str += "Fsm\n";
      str += "States\n";
      for (int j = 0; j < module.fsm.states.size(); j++) {
        FsmState state = module.fsm.states.get(j);
        str += "  " + state.name + "\n";
        for (int k = 0; k < state.transitions.size(); k++) {
          str += "  " + state.transitions.get(k).getSfgsString();
          str += " -->" + state.transitions.get(k).nextState.name
              + "\n";
        }
      }
      str += "\n";
      str += "\n";
    }
    return str;
  }
}
```

B.2.10 Module.java

```
package maaoha.model;

import java.util.ArrayList;
import java.util.StringTokenizer;

import maaoha.MaaohaException;

/*
 * Class responsible for representation of module in the model
 * structure
 */
public class Module {
  protected String name;

  protected Datapath datapath;

  protected Fsm fsm;

  protected Model parentModel;

  /*
   * Creates an instance of the class
   */
  public Module(String name, Model parentModel) {
    this.parentModel = parentModel;
    this.name = name;
    this.datapath = new Datapath(this);
    this.fsm = new Fsm();
  }

  /*
   * Adds transition to the controller
   */
  public FsmTransition addFsmTransition(FsmState source, String strCondition,
      String strSfgs, FsmState target) throws MaaohaException {
    ArrayList<Sfg> sfgs = getSfgsFromString(strSfgs);
    removeRedudantFromList(sfgs);
    Condition condition = new Condition(strCondition, datapath
        .getElements(), datapath.getLookupTables());
    return fsm.addTransitionToState(source, condition, sfgs, target);
  }

  /*
   * Parses transition from string and adds it to the controller
   */
  public void addFsmTransitionFromString(String stateName,
```

```
    String conditionValue, String usedSfgs, String nextStateName)
    throws MaaohaException {
  FsmState state = null;
  for (int i = 0; i < fsm.states.size(); i++) {
    if (fsm.states.get(i).name.equals(stateName))
      state = fsm.states.get(i);
  }

  if (state == null)
    throw new MaaohaException(
        "Module->addFsmTransitionFromString::
           There is no state of given name in the fsm of this module!");

  Condition condition = new Condition(conditionValue, datapath
      .getElements(), datapath.getLookupTables());
  ArrayList<Sfg> sfgs = getSfgsFromString(usedSfgs);

  if (sfgs.size() < 1)
    throw new MaaohaException(
        "Module->addFsmTransitionFromString::There are no signal flow graphs!");

  FsmState nextState = null;

  for (int i = 0; i < fsm.states.size(); i++)
    if (fsm.states.get(i).name.equalsIgnoreCase(nextStateName))
      nextState = fsm.states.get(i);

  if (nextState == null)
    throw new MaaohaException("Invalid next state!");

  fsm.addTransitionToState(state, condition, sfgs, nextState);
}

/*
 * Returns datapath
 */
public Datapath getDatapath() {
  return datapath;
}

/*
 * Returns controller
 */
public Fsm getFsm() {
  return fsm;
}

/*
 * Returns name
 */
public String getName() {
  return name;
}

/*
 * Returns parent model
 */
public Model getParentModel() {
  return parentModel;
}

/*
 * Parses sfgs from string
 */
public ArrayList<Sfg> getSfgsFromString(String strSfgs) {

  ArrayList<Sfg> sfgs = new ArrayList<Sfg>();
  strSfgs = strSfgs.replaceAll("\\(", " ( ");
```

```
    strSfgs = strSfgs.replaceAll("\\)", " ) ");
    strSfgs = strSfgs.replaceAll(",", " , ");
    StringTokenizer tokenizer = new StringTokenizer(strSfgs);
    while (tokenizer.hasMoreTokens()) {
      String token = tokenizer.nextToken();
      for (int i = 0; i < datapath.sfgs.size(); i++) {
        if (datapath.sfgs.get(i).name.equalsIgnoreCase(token))
          sfgs.add(datapath.sfgs.get(i));
      }
    }
    return sfgs;
  }

  /*
   * Removes duplicate entries in the list of sfg
   */
  public void removeRedudantFromList(ArrayList<Sfg> list) {
    for (int i = 0; i < list.size(); i++) {
      for (int j = i + 1; j < list.size(); j++)
        if (list.get(i) == list.get(j)) {
          list.remove(j);
          j--;
        }
    }
  }

  /*
   * Sets controller
   */
  public void setFsm(Fsm fsm) {
    this.fsm = fsm;
  }

  /*
   * Sets name
   */
  public void setName(String name) {
    this.name = name;
  }
}
```

B.2.11 Sfg.java

```
package maaoha.model;

import java.util.ArrayList;

/*
 * Class responsible for representation of signal flow graph
 * in the model structure
 */
public class Sfg {
  protected String name;

  protected ArrayList<Expression> expressions;

  /*
   * Creates instance of the class
   */
  public Sfg(String name) {
    this.name = name;
    expressions = new ArrayList<Expression>();
  }

  /*
   * Creates instance of the class
   */
```

```
public Sfg(String name, ArrayList<Expression> expressions) {
  this.name = name;
  this.expressions = expressions;
}

/*
 * Adds single expression
 */
public void addExpression(Expression expression) {
  expressions.add(expression);
}

/*
 * Returns index of the expression that calculates value for given element
 */
public int doesCalculate(DatapathElement de) {
  for (int i = 0; i < expressions.size(); i++) {
    if (expressions.get(i).leftSideElement == de)
      return i;
  }
  return -1;
}

/*
 * Returns i-th expression
 */
public Expression getExpression(int i) {
  return expressions.get(i);
}

/*
 * Returns number of expressions
 */
public int getExpressionsNo() {
  return expressions.size();
}

/*
 * Returns name
 */
public String getName() {
  return name;
}

/*
 * Returns total number of expressions used on the right side
 */
public int getTotalRightSideElementsNo() {
  int sum = 0;

  for (int j = 0; j < getExpressionsNo(); j++)
    sum += getExpression(j).getRightSideElementsNo();

  return sum;
}

/*
 * Removes expression
 */
public void removeExpression(Expression exp) {
  expressions.remove(exp);
}

/*
 * Sets list of expressions
 */
public void setExpressions(ArrayList<Expression> expressions) {
  this.expressions = expressions;
```

```
  }

  /*
   * Sets name
   */
  public void setName(String name) {
    this.name = name;
  }
}
```

B.2.12 UseModule.java

```
package maaoha.model;

import java.util.ArrayList;

import maaoha.MaaohaException;

/*
 * Class responsible for representation of sub-module
 * in the model structure
 */
public class UseModule {
  protected Module module;

  protected ArrayList<DatapathElement> parameters;

  protected String name;

  /*
   * Creates an instance of the class
   */
  public UseModule(String name, Module module,
      ArrayList<DatapathElement> parameters) throws MaaohaException {
    System.out.println(module.getName());
    if (module.getDatapath().getInputsNo()
        + module.getDatapath().getOutputsNo() == parameters.size()) {
      this.module = module;
      this.parameters = parameters;
      this.name = name;
    } else
      throw new MaaohaException(
          "UseModule->UseModule::Wrong parameter number");
  }

  /*
   * Checks whether all parameters are defined
   */
  public boolean areAllParametersDefined() {
    for (int i = 0; i < parameters.size(); i++)
      if (parameters.get(i) == null)
        return false;
    return true;
  }

  /*
   * Returns sub-module type
   */
  public Module getModule() {
    return module;
  }

  /*
   * Returns name
   */
  public String getName() {
    return name;
```

```
}

/*
 * Returns i-th parameter
 */
public DatapathElement getParameter(int i) {
  return parameters.get(i);
}

/*
 * Returns parameters number
 */
public int getParametersNo() {
  return parameters.size();
}

/*
 * Returns type of i-th parameter
 */
public String getTypeOfParameter(int i) throws MaaohaException {
  return module.datapath.getInputOutputType(i);
}

/*
 * Returns parameters as string
 */
public String parametersString() {
  String res = "";

  if (parameters.size() > 0)
    res += parameters.get(0).name;
  for (int i = 1; i < parameters.size(); i++)
    res += "," + parameters.get(i).name;

  return res;
}

/*
 * Removes parameter
 */
public void removeParameter(int index) {
  parameters.remove(index);
}

/*
 * Sets sub-module type
 */
public void setModule(Module module) {
  this.module = module;
}

/*
 * Sets name
 */
public void setName(String name) {
  this.name = name;
}

/*
 * Sets i-th parameter
 */
public void setParameter(int i, DatapathElement de) {
  parameters.remove(i);
  parameters.add(i, de);
}

/*
 * Sets parameters
```

```
   */
  public void setParameters(ArrayList<DatapathElement> parameters) {
    this.parameters = parameters;
  }
}
```

B.2.13 UseModuleAndParameter.java

```
package maaoha.model;

/*
 * Auxiliary class that associates module instance
 * with element that acts as its parameter
 */
public class UseModuleAndParameter {
  public Module module;

  public DatapathElement element;
  public UseModule useModule;

  /*
   * Creates an instance of the class
   */
  public UseModuleAndParameter(DatapathElement element, UseModule useModule,
      Module module) {
    this.element = element;
    this.useModule = useModule;
    this.module = module;
  }
}
```

B.3 Package maaoha.model.expression

B.3.1 BasicExpression.java

```
package maaoha.model.expression;

import java.util.ArrayList;

import maaoha.MaaohaException;
import maaoha.model.DatapathElement;
import maaoha.model.LookUpTable;

/*
 * Parent class for all classes that represents elements of expressions
 */
public class BasicExpression {
  final public static int Brackets = 10;
  final public static int BitwiseAnd = 12;
  final public static int BitwiseOr = 13;
  final public static int TernaryConditional = 14;
  final public static int ShiftLeft = 15;
  final public static int ShiftRight = 16;
  final public static int Addition = 17;
  final public static int Subtraction = 18;
  final public static int ConstantValue = 19;
  final public static int DatapathElement = 20;
  final public static int Equality = 21;
  final public static int LessOrEqual = 22;
  final public static int GreaterOrEqual = 23;
  final public static int Less = 24;
  final public static int Greater = 25;
  final public static int BitSelection = 26;
  final public static int Negation = 27;
```

```
final public static int LookupOperator = 28;
final public static int BitwiseXor = 29;
final public static int Inequality = 30;
final public static int Concatenation = 31;
final public static int LookupOperand = 32;

/*
 * Checks whether token is within brackets
 */
static protected boolean betweenSquareBrackets(String expression, int pos)
    throws MaaohaException {
  int begin = 0;
  int end = 0;

  while (true) {
    begin = expression.indexOf('[', begin + 1);
    end = expression.indexOf(']', end + 1);

    if (begin == -1 && end == -1)
      return false;
    else if ((begin == -1 && end != -1) || (begin != -1 && end == -1))
      throw new MaaohaException(
          "BasicExpression->betweenSquareBrackets::unmached square bracket!");
    else if (pos < begin)
      return false;
    else if (pos < end)
      return true;
  }
}

/*
 * Parses string and creates basic expression
 */
static public BasicExpression createBasicExpression(String expression,
    ArrayList<DatapathElement> datapathElements,
    ArrayList<LookUpTable> lookupTables) throws MaaohaException {
  int firstBracket = expression.indexOf('(');
  ArrayList<String> removedStrings = new ArrayList<String>();

  while (firstBracket != -1) {
    int index = firstBracket + 1;
    int numberOfBrackets = 1;
    while (numberOfBrackets > 0) {
      if (index <= expression.length()) {
        if (expression.charAt(index) == '(') {
          numberOfBrackets++;
        } else if (expression.charAt(index) == ')') {
          numberOfBrackets--;
        }
      } else
        throw new MaaohaException(
            "BasicExpression->BasicExpression: Parsing error: brackets!");
      index++;
    }
    removedStrings.add(expression.substring(firstBracket, index));
    expression = expression.substring(0, firstBracket) + "string_"
        + removedStrings.size() + expression.substring(index);

    firstBracket = expression.indexOf('(');
  }

  int charPos = expression.indexOf(" ? ");
  if (charPos > -1) {
    int charPos2 = expression.indexOf(" : ");

    while (betweenSquareBrackets(expression, charPos2))
      charPos2 = expression.indexOf(" : ", charPos2 + 1);
```

```
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3, charPos2);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  String exp3 = expression.substring(charPos2 + 3);
  exp3 = restoreReplacedStrings(exp3, removedStrings);
  return new TernaryOperator(
      TernaryConditional,
      createBasicExpression(exp1, datapathElements, lookupTables),
      createBasicExpression(exp2, datapathElements, lookupTables),
      createBasicExpression(exp3, datapathElements, lookupTables));
}
charPos = expression.indexOf(" | ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(BitwiseOr, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" ^ ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(BitwiseXor, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" & ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(BitwiseAnd, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" # ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(Concatenation, createBasicExpression(
      exp1, datapathElements, lookupTables),
      createBasicExpression(exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf("~ ");
if (charPos > -1) {
  String exp1 = expression.substring(charPos + 1);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  return new UnaryOperator(Negation, createBasicExpression(exp1,
      datapathElements, lookupTables));
}
charPos = expression.indexOf(" == ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 4);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(Equality, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
```

```
}
charPos = expression.indexOf(" != ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 4);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(Inequality, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" < ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(Less, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" > ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(Greater, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" <= ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 4);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(LessOrEqual, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" >= ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 4);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(GreaterOrEqual, createBasicExpression(
      exp1, datapathElements, lookupTables),
      createBasicExpression(exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" << ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 4);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(ShiftLeft, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" >> ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 4);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(ShiftRight, createBasicExpression(exp1,
```

```
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" + ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(Addition, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf(" - ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = restoreReplacedStrings(exp1, removedStrings);
  String exp2 = expression.substring(charPos + 3);
  exp2 = restoreReplacedStrings(exp2, removedStrings);
  return new BinaryOperator(Subtraction, createBasicExpression(exp1,
      datapathElements, lookupTables), createBasicExpression(
      exp2, datapathElements, lookupTables));
}
charPos = expression.indexOf("string");
if (charPos > -1) {
  String lookup = expression.substring(0, charPos);
  lookup = lookup.replaceAll(" ", "");
  LookUpTable table = getLookupTable(lookupTables, lookup);
  if (table != null) {
    String exp1 = expression.substring(charPos);
    exp1 = restoreReplacedStrings(exp1, removedStrings);
    exp1 = exp1.replaceAll("\\(", "");
    exp1 = exp1.replaceAll("\\)", "");

    return new BinaryOperator(LookupOperator,
        new LookupTableOperator(table), createBasicExpression(
            exp1, datapathElements, lookupTables));
  }
}
charPos = expression.indexOf(" [ ");
if (charPos > -1) {
  String exp1 = expression.substring(0, charPos);
  exp1 = exp1.replaceAll(" ", "");
  DatapathElement de = null;
  for (int i = 0; i < datapathElements.size(); i++) {
    if (datapathElements.get(i).getName().equals(exp1)) {
      de = datapathElements.get(i);
      i = datapathElements.size();
    }
  }
  if (de == null)
    throw new MaaohaException(
        "BasicExpression->createBasicExpression::Unknown variable: "
            + exp1);

  int charPos2 = expression.indexOf(" : ");
  if (charPos2 > -1) {
    String exp2 = expression.substring(charPos + 2, charPos2);
    exp2 = exp2.replaceAll(" ", "");
    int number1;
    try {
      number1 = Integer.parseInt(exp2);
    } catch (NumberFormatException ex) {
      throw new MaaohaException(
          "BasicExpression->createBasicExpression::
              Syntax error, can't parse number :"
              + exp2);
    }
```

```
      charPos = charPos2;
      charPos2 = expression.indexOf(" ]");
      if (charPos2 > -1) {
        exp2 = expression.substring(charPos + 2, charPos2);
        exp2 = exp2.replaceAll(" ", "");
        try {
          int number2 = Integer.parseInt(exp2);
          return new BitSelection(BitSelection, de, number1,
              number2);
        } catch (NumberFormatException ex) {
          throw new MaaohaException(
              "BasicExpression->createBasicExpression::
                   Syntax error, can't parse number :"
                  + exp2);
        }
      } else
        throw new MaaohaException(
            "BasicExpression->createBasicExpression::
                Syntax error, no closing ']' in :"
                + expression);
    } else {
      charPos2 = expression.indexOf(" ]");
      if (charPos2 > -1) {
        String exp2 = expression.substring(charPos + 2, charPos2);
        exp2 = exp2.replaceAll(" ", "");
        try {
          int number = Integer.parseInt(exp2);
          return new BitSelection(BitSelection, de, number,
              number);
        } catch (NumberFormatException ex) {
          throw new MaaohaException(
              "BasicExpression->createBasicExpression::
                  Syntax error, can't parse number :"
                  + exp2);
        }
      } else
        throw new MaaohaException(
            "BasicExpression->createBasicExpression::
                Syntax error, no closing ']' in :"
                + expression);
    }
  }
  expression = expression.replaceAll(" ", "");
  int indexOfString = expression.indexOf("string_");
  if (indexOfString > -1) {
    int indexOfSpace = expression.indexOf(" ", indexOfString);
    int number = -1;

    if (indexOfSpace < 0) {
      number = Integer.parseInt(expression
          .substring(indexOfString + 7));
      expression = expression.substring(0, indexOfString)
          + removedStrings.get(number - 1);
    } else {
      number = Integer.parseInt(expression.substring(
          indexOfString + 7, indexOfSpace));
      expression = expression.substring(0, indexOfString)
          + removedStrings.get(number - 1)
          + expression.substring(indexOfSpace);
    }

    int indexOfFirstBracket = expression.indexOf("(");
    int indexOfLastBracket = -1;
    int index = expression.indexOf(")");
    while (index > -1) {
      indexOfLastBracket = index;
      index = expression.indexOf(")", index + 1);
```

```
    }
    return new UnaryOperator(Brackets, createBasicExpression(expression
        .substring(indexOfFirstBracket + 1, indexOfLastBracket),
        datapathElements, lookupTables));
  }

  try {
    int number;
    if (expression.length() > 1
        && expression.substring(0, 2).equalsIgnoreCase("0x")) {
      expression = expression.substring(2);
      number = Integer.parseInt(expression, 16);
    } else {
      number = Integer.parseInt(expression);
    }
    return new Constant(ConstantValue, number);
  } catch (NumberFormatException ex) {
    for (int i = 0; i < datapathElements.size(); i++) {
      if (datapathElements.get(i).getName().equals(expression)) {
        return new Variable(DatapathElement, datapathElements
            .get(i));
      }
    }
  }
  throw new MaaohaException(
      "BasicExpression->createBasicExpression::Returns null");
}

/*
 * Returns lookup table
 */
static protected LookUpTable getLookupTable(ArrayList<LookUpTable> tables,
    String name) {
  for (int i = 0; i < tables.size(); i++) {
    if (tables.get(i).getName().equals(name))
      return tables.get(i);
  }
  return null;
}

/*
 * Restores values replaced by 'string_index'
 */
static protected String restoreReplacedStrings(String exp,
    ArrayList<String> removedStrings) {
  int indexOfString = exp.indexOf("string_");
  while (indexOfString > -1) {
    int number;
    int indexOfSpace = exp.indexOf(" ", indexOfString);
    if (indexOfSpace < 0) {
      number = Integer.parseInt(exp.substring(indexOfString + 7));
      exp = exp.substring(0, indexOfString)
          + removedStrings.get(number - 1);
    } else {
      number = Integer.parseInt(exp.substring(indexOfString + 7,
          indexOfSpace));
      exp = exp.substring(0, indexOfString)
          + removedStrings.get(number - 1)
          + exp.substring(indexOfSpace);
    }

    indexOfString = exp.indexOf("string_");
  }
  return exp;
}

protected int type;
```

```
  /*
   * Constructor
   */
  public BasicExpression() {
  }

  /*
   * (non-Javadoc)
   *
   * @see java.lang.Object#clone()
   */
  public BasicExpression clone() {
    BasicExpression be = new BasicExpression();
    return be;
  }

  /*
   * Checks whether element of given type is present
   */
  public boolean containsExpressionOfType(int type) {
    if (this.type == type)
      return true;

    return false;
  }

  /*
   * Returns number of elements of specified type inside expression
   */
  public int getNoOfExpressionsOfType(int type) {
    if (this.type == type)
      return 1;
    return 0;
  }

  /*
   * Returns type of the expression element
   */
  public int getType() {
    return type;
  }

  /*
   * Returns value of the expression element
   */
  public int getValueType() throws MaaohaException {
    throw new MaaohaException("BasicExpression->getValueType::Unknown type");
  }

  /*
   * Sets element
   */
  public void setElement(DatapathElement oldElement,
      DatapathElement newElement) {
  }

  public String toString() {
    return "ERROR - toString called from BasicExpression";
  }
}
```

B.3.2 BinaryOperator.java

```
package maaoha.model.expression;

import maaoha.MaaohaException;
import maaoha.model.DatapathElement;
```

```java
/*
 * Class represents binary operators in expression
 */
public class BinaryOperator extends BasicExpression {
  protected BasicExpression expression1;

  protected BasicExpression expression2;

  /*
   * Constructor
   */
  public BinaryOperator(int type, BasicExpression expression1,
      BasicExpression expression2) {
    this.type = type;
    this.expression1 = expression1;
    this.expression2 = expression2;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#clone()
   */
  public BasicExpression clone() {
    BasicExpression expression1 = this.expression1.clone();
    BasicExpression expression2 = this.expression2.clone();
    BinaryOperator be = new BinaryOperator(type, expression1, expression2);
    return be;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#containsExpressionOfType(int)
   */
  public boolean containsExpressionOfType(int type) {
    if (this.type == type)
      return true;
    if (expression1.containsExpressionOfType(type))
      return true;
    if (expression2.containsExpressionOfType(type))
      return true;
    return false;
  }

  /*
   * Returns left expression
   */
  public BasicExpression getExpression1() {
    return expression1;
  }

  /*
   * Returns right expression
   */
  public BasicExpression getExpression2() {
    return expression2;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#getNoOfExpressionsOfType(int)
   */
  public int getNoOfExpressionsOfType(int type) {
    int sum = 0;
    if (this.type == type)
```

```
      sum++;
    sum += expression1.getNoOfExpressionsOfType(type);
    sum += expression2.getNoOfExpressionsOfType(type);
    return sum;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#getValueType()
   */
  public int getValueType() throws MaaohaException {
    if (type == BasicExpression.BitwiseAnd
        || type == BasicExpression.BitwiseOr
        || type == BasicExpression.BitwiseXor
        || type == BasicExpression.ShiftLeft
        || type == BasicExpression.ShiftRight
        || type == BasicExpression.Addition
        || type == BasicExpression.Subtraction) {
      int t1 = expression1.getValueType();
      int t2 = expression2.getValueType();

      if (t1 == t2)
        return t1;
      else if (t1 == -1 || t2 > t1)
        return t2;
      else if (t2 == -1 || t1 > t2)
        return t1;

      throw new MaaohaException(
          "BinaryOperator->getValueType::Invlaid types");
    }
    if (type == BasicExpression.Equality || type == BasicExpression.Greater
        || type == BasicExpression.GreaterOrEqual
        || type == BasicExpression.Less
        || type == BasicExpression.LessOrEqual)
      return 1;
    if (type == BasicExpression.ShiftRight
        || type == BasicExpression.ShiftLeft)
      return expression1.getValueType();
    if (type == BasicExpression.Concatenation)
      return expression1.getValueType() + expression2.getValueType();

    throw new MaaohaException(
        "BinaryOperator->getValueType::Unknown type : " + type);
  }

  /*
   * (non-Javadoc)
   *
   * @see
maaoha.model.expression.BasicExpression#setElement(maaoha.model.DatapathElement,
   *      maaoha.model.DatapathElement)
   */
  public void setElement(DatapathElement oldElement,
      DatapathElement newElement) {
    expression1.setElement(oldElement, newElement);
    expression2.setElement(oldElement, newElement);
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#toString()
   */
  public String toString() {
    if (type == BasicExpression.Addition)
      return (expression1 + " + " + expression2);
```

```
    if (type == BasicExpression.BitwiseAnd)
      return (expression1 + " & " + expression2);
    if (type == BasicExpression.BitwiseOr)
      return (expression1 + " | " + expression2);
    if (type == BasicExpression.BitwiseXor)
      return (expression1 + " ^ " + expression2);
    if (type == BasicExpression.Equality)
      return (expression1 + " == " + expression2);
    if (type == BasicExpression.Inequality)
      return (expression1 + " != " + expression2);
    if (type == BasicExpression.Greater)
      return (expression1 + " > " + expression2);
    if (type == BasicExpression.GreaterOrEqual)
      return (expression1 + " >= " + expression2);
    if (type == BasicExpression.Less)
      return (expression1 + " < " + expression2);
    if (type == BasicExpression.LessOrEqual)
      return (expression1 + " <= " + expression2);
    if (type == BasicExpression.ShiftLeft)
      return (expression1 + " << " + expression2);
    if (type == BasicExpression.ShiftRight)
      return (expression1 + " >> " + expression2);
    if (type == BasicExpression.Subtraction)
      return (expression1 + " - " + expression2);
    if (type == BasicExpression.LookupOperator)
      return (expression1 + "(" + expression2 + ")");
    if (type == BasicExpression.Concatenation)
      return (expression1 + " # " + expression2);
    return "unknown type: " + type;
  }
}
```

B.3.3 BitSelecetion.java

```
package maaoha.model.expression;

import maaoha.MaaohaException;
import maaoha.model.DatapathElement;

/*
 * Class represents bit selection in expression
 */
public class BitSelection extends BasicExpression {
  protected DatapathElement element;

  protected int firstBit;

  protected int lastBit;

  /*
   * Constructor
   */
  public BitSelection(int type, DatapathElement element, int firstBit,
      int lastBit) {
    this.type = type;
    this.element = element;
    this.firstBit = firstBit;
    this.lastBit = lastBit;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#clone()
   */
  public BasicExpression clone() {
    BitSelection be = new BitSelection(type, element, firstBit, lastBit);
```

```
    return be;
  }

  /*
   * Returns datapath element
   */
  public DatapathElement getElement() {
    return element;
  }

  /*
   * Returns index of the first bit
   */
  public int getFirstBit() {
    return firstBit;
  }

  /*
   * Returns index of the last bit
   */
  public int getLastBit() {
    return lastBit;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#getValueType()
   */
  public int getValueType() throws MaaohaException {
    return lastBit - firstBit + 1;
  }

  /*
   * (non-Javadoc)
   *
   * @see
maaoha.model.expression.BasicExpression#setElement(maaoha.model.DatapathElement,
   *      maaoha.model.DatapathElement)
   */
  public void setElement(DatapathElement oldElement,
      DatapathElement newElement) {
    if (element == oldElement)
      element = newElement;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#toString()
   */
  public String toString() {
    if (type == BasicExpression.BitSelection && element != null)
      return (element.getName() + "[" + firstBit + ":" + lastBit + "]");

    return "dziwy" + type;
  }
}
```

B.3.4 Constant.java

```
package maaoha.model.expression;

/*
 * Class represents constant in expression
 */
public class Constant extends BasicExpression {
```

```
  protected int value;

  /*
   * Constructor
   */
  public Constant(int value) {
    this.type = BasicExpression.ConstantValue;
    this.value = value;
  }

  /*
   * Constructor
   */
  public Constant(int type, int value) {
    this.type = type;
    this.value = value;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#clone()
   */
  public BasicExpression clone() {
    Constant be = new Constant(type, value);
    return be;
  }

  /*
   * Returns value
   */
  public int getValue() {
    return value;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#getValueType()
   */
  public int getValueType() {
    int number = 0;
    int val = value;
    while (val > 0) {
      number++;
      val /= 2;
    }
    if (number == 0)
      number = 1;
    return number;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#toString()
   */
  public String toString() {
    return "" + value;
  }
}
```

B.3.5 LookupTableOperator.java

```
package maaoha.model.expression;

import maaoha.MaaohaException;
```

```
import maaoha.model.LookUpTable;

/*
 * Class represents lookup table in expression
 */
public class LookupTableOperator extends BasicExpression {
  protected LookUpTable table;

  /*
   * Constructor
   */
  public LookupTableOperator(LookUpTable table) {
    this.table = table;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#clone()
   */
  public BasicExpression clone() {
    LookupTableOperator be = new LookupTableOperator(table);
    return be;
  }

  /*
   * Returns lookup table
   */
  public LookUpTable getTable() {
    return table;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#getValueType()
   */
  public int getValueType() throws MaaohaException {
    String type = table.getType();
    int charPos = type.indexOf("ns(");
    int charPos2 = type.indexOf(")");
    if (charPos == -1 || charPos2 == -1)
      throw new MaaohaException(
          "Variable->getValueType: Cannot parse type to number");
    int number = Integer.parseInt(type.substring(charPos + 3, charPos2));
    return number;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#toString()
   */
  public String toString() {
    return table.getName();
  }
}
```

B.3.6 TernaryOperator.java

```
package maaoha.model.expression;

import maaoha.model.DatapathElement;

/*
 * Class represents ternary operators in expression
```

```
 */
public class TernaryOperator extends BasicExpression {
  protected BasicExpression expression1;

  protected BasicExpression expression2;

  protected BasicExpression expression3;

  /*
   * Constructor
   */
  public TernaryOperator(int type, BasicExpression expression1,
      BasicExpression expression2, BasicExpression expression3) {
    this.type = type;
    this.expression1 = expression1;
    this.expression2 = expression2;
    this.expression3 = expression3;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#clone()
   */
  public BasicExpression clone() {
    BasicExpression expression1 = this.expression1.clone();
    BasicExpression expression2 = this.expression2.clone();
    BasicExpression expression3 = this.expression3.clone();
    TernaryOperator be = new TernaryOperator(type, expression1,
        expression2, expression3);
    return be;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#containsExpressionOfType(int)
   */
  public boolean containsExpressionOfType(int type) {
    if (this.type == type)
      return true;
    if (expression1.containsExpressionOfType(type))
      return true;
    if (expression2.containsExpressionOfType(type))
      return true;
    if (expression3.containsExpressionOfType(type))
      return true;
    return false;
  }

  /*
   * Return left expression
   */
  public BasicExpression getExpression1() {
    return expression1;
  }

  /*
   * Returns middle expression
   */
  public BasicExpression getExpression2() {
    return expression2;
  }

  /*
   * Returns right expression
   */
  public BasicExpression getExpression3() {
```

```
    return expression3;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#getNoOfExpressionsOfType(int)
   */
  public int getNoOfExpressionsOfType(int type) {
    int sum = 0;
    if (this.type == type)
      sum++;
    sum += expression1.getNoOfExpressionsOfType(type);
    sum += expression2.getNoOfExpressionsOfType(type);
    sum += expression3.getNoOfExpressionsOfType(type);
    return sum;
  }

  /*
   * (non-Javadoc)
   *
   * @see
maaoha.model.expression.BasicExpression#setElement(maaoha.model.DatapathElement,
   *      maaoha.model.DatapathElement)
   */
  public void setElement(DatapathElement oldElement,
      DatapathElement newElement) {
    expression1.setElement(oldElement, newElement);
    expression2.setElement(oldElement, newElement);
    expression3.setElement(oldElement, newElement);
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#toString()
   */
  public String toString() {
    if (type == BasicExpression.TernaryConditional)
      return (expression1 + " ? " + expression2 + " : " + expression3);
    return "unknown type" + type;
  }
}
```

B.3.7 UnaryOperator.java

```
package maaoha.model.expression;

import maaoha.MaaohaException;
import maaoha.model.DatapathElement;

/*
 * Class represents unary operators in expression
 */
public class UnaryOperator extends BasicExpression {
  protected BasicExpression expression;

  /*
   * Constructors
   */
  public UnaryOperator(int type, BasicExpression expression) {
    this.type = type;
    this.expression = expression;
  }

  /*
   * (non-Javadoc)
```

```
     *
     * @see maaoha.model.expression.BasicExpression#clone()
     */
    public BasicExpression clone() {
      BasicExpression expression = this.expression.clone();
      UnaryOperator be = new UnaryOperator(type, expression);
      return be;
    }

    /*
     * (non-Javadoc)
     *
     * @see maaoha.model.expression.BasicExpression#containsExpressionOfType(int)
     */
    public boolean containsExpressionOfType(int type) {
      if (expression.containsExpressionOfType(type))
        return true;
      return false;
    }

    /*
     * Returns expression
     */
    public BasicExpression getExpression() {
      return expression;
    }

    /*
     * (non-Javadoc)
     *
     * @see maaoha.model.expression.BasicExpression#getNoOfExpressionsOfType(int)
     */
    public int getNoOfExpressionsOfType(int type) {
      int sum = 0;
      if (this.type == type)
        sum++;
      sum += expression.getNoOfExpressionsOfType(type);
      return sum;
    }

    /*
     * (non-Javadoc)
     *
     * @see maaoha.model.expression.BasicExpression#getValueType()
     */
    public int getValueType() throws MaaohaException {
      return expression.getValueType();
    }

    /*
     * (non-Javadoc)
     *
     * @see
  maaoha.model.expression.BasicExpression#setElement(maaoha.model.DatapathElement,
     *      maaoha.model.DatapathElement)
     */
    public void setElement(DatapathElement oldElement,
        DatapathElement newElement) {
      expression.setElement(oldElement, newElement);
    }

    /*
     * (non-Javadoc)
     *
     * @see maaoha.model.expression.BasicExpression#toString()
     */
    public String toString() {
      if (type == BasicExpression.Brackets)
```

```
      return ("(" + expression + ")");
    if (type == BasicExpression.Negation)
      return ("~" + expression);

    return "unknown type: " + type;
  }
}
```

B.3.8 Variable.java

```
package maaoha.model.expression;

import maaoha.MaaohaException;
import maaoha.model.DatapathElement;

/*
 * Class represents variables in expression
 */
public class Variable extends BasicExpression {
  protected DatapathElement element;

  /*
   * Constructor
   */
  public Variable(int type, DatapathElement element) {
    this.type = type;
    this.element = element;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#clone()
   */
  public BasicExpression clone() {
    Variable be = new Variable(type, element);
    return be;
  }

  /*
   * Returns element
   */
  public DatapathElement getElement() {
    return element;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.BasicExpression#getValueType()
   */
  public int getValueType() throws MaaohaException {
    String type = element.getValueType();
    int charPos = type.indexOf("ns(");
    int charPos2 = type.indexOf(")");
    if (charPos == -1 || charPos2 == -1)
      throw new MaaohaException(
          "Variable->getValueType: Cannot parse type to number");
    int number = Integer.parseInt(type.substring(charPos + 3, charPos2));

    return number;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.model.expression.
```

```
     *       BasicExpression#setElement(maaoha.model.DatapathElement,
     *       maaoha.model.DatapathElement)
     */
    public void setElement(DatapathElement oldElement,
        DatapathElement newElement) {
      if (element == oldElement)
        element = newElement;
    }

    /*
     * (non-Javadoc)
     *
     * @see maaoha.model.expression.BasicExpression#toString()
     */
    public String toString() {
      if (element != null)
        return element.getName();
      else
        return "???";
    }
}
```

B.4 Package maaoha.source

B.4.1 GezelGenerator.java

```
package maaoha.source;

import maaoha.model.DatapathElement;
import maaoha.model.Fsm;
import maaoha.model.FsmState;
import maaoha.model.FsmTransition;
import maaoha.model.Model;
import maaoha.model.Module;
import maaoha.model.Sfg;
import maaoha.model.UseModule;

/*
 * Class responsible for generation of GEZEL program
 */
public class GezelGenerator {
  static public String eol = "\n";

  /*
   * Generates GEZEL program
   */
  static public String generateGezelCode(Model model) {
    String program = "";

    for (int i = 0; i < model.getModulesNo(); i++) {
      Module module = model.getModule(i);

      // module name
      program += "dp " + module.getName();
      // parameters
      if (module.getDatapath().getInputsNo() > 0
          || module.getDatapath().getOutputsNo() > 0) {
        program += "(";
        for (int j = 0; j < module.getDatapath().getElementsNo(); j++) {
          DatapathElement de = module.getDatapath().getElement(j);
          if (de.getElementType().equals(Model.typeInput)) {
            program += "in " + de.getName() + " : "
                + de.getValueType() + ";" + eol;
          } else if (de.getElementType().equals(Model.typeOutput)) {
            program += "out " + de.getName() + " : "
                + de.getValueType() + ";" + eol;
```

```
    }

  }
  program += ")";
}

program += eol + eol + "{" + eol;

// variables declaration
for (int j = 0; j < module.getDatapath().getElementsNo(); j++) {
  DatapathElement de = module.getDatapath().getElement(j);
  if (de.getElementType().equals(Model.typeSignal)) {
    program += "sig " + de.getName() + " : "
        + de.getValueType() + ";" + eol;
  } else if (de.getElementType().equals(Model.typeRegister)) {
    program += "reg " + de.getName() + " : "
        + de.getValueType() + ";" + eol;
  }
}

program += eol;

// signal flow graphs definition
for (int j = 0; j < module.getDatapath().getSfgsNo(); j++) {
  Sfg sfg = module.getDatapath().getSfg(j);

  program += "sfg " + sfg.getName() + " { ";
  for (int k = 0; k < sfg.getExpressionsNo(); k++) {
    program += sfg.getExpression(k).getValue() + "; ";
  }
  program += " } " + eol;
}

// use of sub-modules
for (int j = 0; j < module.getDatapath().getUseModulesNo(); j++) {
  UseModule um = module.getDatapath().getUseModule(j);
  program += "use " + um.getModule().getName() + "(";
  for (int k = 0; k < um.getParametersNo(); k++) {
    program += um.getParameter(k).getName();
    if (k < um.getParametersNo() - 1)
      program += ",";
  }
  program += ");" + eol;
}

program += "}" + eol + eol;

// controler definition
program += "fsm " + module.getName() + "_ctl(" + module.getName()
    + ") {" + eol;
Fsm fsm = module.getFsm();
// state declarations
for (int j = 0; j < fsm.getStatesNo(); j++) {
  FsmState fs = fsm.getFsmState(j);
  if (fsm.getInitialState() != null
      && fsm.getInitialState().equals(fs))
    program += "initial " + fs.getName() + ";" + eol;
  else
    program += "state " + fs.getName() + ";" + eol;
}
program += eol;

// transition declartions
for (int j = 0; j < fsm.getStatesNo(); j++) {
  FsmState fs = fsm.getFsmState(j);
  for (int k = 0; k < fs.getTransitionsNo(); k++) {
    if (k == 0)
      program += "@" + fs.getName();
```

```
          else
            program += "    ";

          FsmTransition ft = fs.getTransistion(k);
          if (ft.getCondition().getBasicExpression().toString()
              .equals("1")
              && k == fs.getTransitionsNo() - 1 && k != 0) {
            program += "   else ";
          }
          if (!ft.getCondition().getBasicExpression().toString()
              .equals("1")) {
            program += " if "
                + ft.getCondition().getBasicExpression()
                    .toString() + " then";
          }

          program += " (" + ft.getSfgsString() + ") -> "
              + ft.getNextState().getName() + ";" + eol;
        }
      }
    }
    return program;
  }
}
```

B.4.2 GezelParser.java

```
package maaoha.source;

import java.io.BufferedReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.StringTokenizer;

import maaoha.MaaohaException;
import maaoha.model.Condition;
import maaoha.model.DatapathElement;
import maaoha.model.Expression;
import maaoha.model.Fsm;
import maaoha.model.FsmState;
import maaoha.model.Model;
import maaoha.model.Module;
import maaoha.model.Sfg;
import maaoha.model.UseModule;
import maaoha.tools.MaaohaTools;

/*
 * Class provides parsing of GEZEL programs
 * into the tool internal data structure
 */
public class GezelParser {
  /*
   * Temporary structure for instances of modules
   */
  class ModuleCopy {
    String name;

    Module module;

    public ModuleCopy(String name, Module module) {
      this.name = name;
      this.module = module;
    }
  }

  ArrayList<ModuleCopy> moduleCopies = new ArrayList<ModuleCopy>();
```

```
private String line;

private BufferedReader in;

private StringTokenizer tokenizer;

private int lineNo;

/*
 * Returns module type of the specified module instance
 */
protected Module findModuleCopy(String name, ArrayList<ModuleCopy> copies) {
  for (int i = 0; i < copies.size(); i++)
    if (copies.get(i).name.equals(name))
      return copies.get(i).module;
  return null;
}

/*
 * Returns next token from the GEZEL program
 */
private String getNextToken(boolean possibleEnd) throws IOException {
  if (readNewLine(possibleEnd))
    return null;

  String token = tokenizer.nextToken();
  ;

  // skip comments
  if (token.equals("//")) {
    line = null;
    tokenizer = null;
    if (readNewLine(possibleEnd))
      return null;

    return getNextToken(possibleEnd);
  }
  // return next token
  return token;
}

/*
 * Parses sfg 'always'
 */
private void parseAlways(Module module) throws IOException, MaaohaException {
  String sfgName = "always";

  // creating corresponding sfg
  Sfg sfg = new Sfg(sfgName);
  parseSfgExpressions(sfg, module);

  // creating simple fsm
  Fsm fsm = new Fsm();
  FsmState state = fsm.createNewState("s0");
  ArrayList<Sfg> sfgs = new ArrayList<Sfg>();
  sfgs.add(sfg);
  fsm.addTransitionToState(state, new Condition("1", module.getDatapath()
      .getElements(), module.getDatapath().getLookupTables()), sfgs,
      state);
  module.setFsm(fsm);
}

/*
 * Parses datapath elements
 */
private void parseDpElements(String elType, Module module)
    throws IOException {
  ArrayList<String> ll = new ArrayList<String>();
```

```
  String token = getNextToken(false);
  while (!token.equalsIgnoreCase(":")) {
    if (token.equalsIgnoreCase(",")) {
      token = getNextToken(false);
    }
    ll.add(token);
    token = getNextToken(false);
  }
  token = getNextToken(false);
  token = getNextToken(false);
  token = getNextToken(false);
  String valType = token;
  String elem = null;
  if (ll.isEmpty())
    elem = null;
  else
    elem = ll.get(0);
  while (elem != null) {
    module.getDatapath().addDatapathElement(
        new DatapathElement(elem, elType, "ns(" + valType + ")"));
    ll.remove(0);
    if (ll.isEmpty())
      elem = null;
    else
      elem = ll.get(0);
  }
  token = getNextToken(false);
}

/*
 * Parses GEZEL program
 */
public void parseGezelCode(BufferedReader in, Model model) {
  try {
    this.in = in;
    line = null;
    tokenizer = null;
    lineNo = 0;

    String token = getNextToken(true);
    while (token != null) {
      if (token.equals("system")) {
        token = getNextToken(false); // s
        token = getNextToken(false); // {
        token = getNextToken(false); // module name
        model.setAsSystemModule(token);
        token = getNextToken(false); // ;
        token = getNextToken(false); // }

      } else if (token.equals("dp")) {
        // Inside datapath
        token = getNextToken(false);
        String moduleName = token;
        token = getNextToken(false);

        if (token.equals(":")) {
          token = getNextToken(false);
          String moduleType = token;
          moduleCopies.add(new ModuleCopy(moduleName, model
              .getModule(moduleType)));
        } else if (token.equals("(") || token.equals("{")) {
          Module module = new Module(moduleName, model);
          if (token.equalsIgnoreCase("(")) {
            token = getNextToken(false);
            while (!token.equalsIgnoreCase(")")) {
              if (token.equalsIgnoreCase("in")) {
                // inputs
                parseDpElements(Model.typeInput, module);
```

```
        } else if (token.equalsIgnoreCase("out")) {
          // outputs
          parseDpElements(Model.typeOutput, module);
        }
        token = getNextToken(false);
      }
      token = getNextToken(false);
    }
    if (token.equalsIgnoreCase("{")) {
      // inside datapath body
      token = getNextToken(false);
      while (!token.equalsIgnoreCase("}")) {
        if (token.equalsIgnoreCase("lookup")) {
          // adding lookup table
          parseLookUpTable(module);
        } else if (token.equalsIgnoreCase("reg")) {
          // adding registers
          parseDpElements(Model.typeRegister, module);
        } else if (token.equalsIgnoreCase("sig")) {
          // adding signals
          parseDpElements(Model.typeSignal, module);
        } else if (token.equalsIgnoreCase("sfg")) {
          // adding signal flow graphs
          parseSfg(module);
        } else if (token.equalsIgnoreCase("use")) {
          // adding use datapath
          parseUseModule(module, model);
        } else if (token.equalsIgnoreCase("always")) {
          // adding use datapath
          parseAlways(module);
        }
        token = getNextToken(false);
      }
    }
    if (module != null)
      model.addModule(module);
  }
} else if (token.equalsIgnoreCase("fsm")) {
  // Inside fsm
  token = getNextToken(false); // name
  Fsm fsm = new Fsm();
  token = getNextToken(false); // "("
  token = getNextToken(false); // datapath name
  String datapathName = token;
  Module module = model.getModule(datapathName);
  model.addFsmToModule(fsm, datapathName);
  token = getNextToken(false); // ")"
  token = getNextToken(false); // "{"
  while (!token.equalsIgnoreCase("}")) {
    if (token.equalsIgnoreCase("initial")) {
      token = getNextToken(false); // state_name
      fsm.setInitialState(fsm.createNewState(token));
      token = getNextToken(false); // ";"
    } else if (token.equalsIgnoreCase("state")) {
      while (!token.equalsIgnoreCase(";")) {
        token = getNextToken(false); // state_name
        fsm.createNewState(token);
        token = getNextToken(false); // "," || ";"
      }
    } else if (token.charAt(0) == '@') {
      token = token.substring(1);
      parseTransition(module, fsm, token);
    }

    token = getNextToken(false); // "initial" || "state"
                    // || "@state_name" ||
                    // "}"
  }
```

```
      } else {
        System.out.println("Gezel file error line = " + lineNo);
        System.exit(1);
      }
      token = getNextToken(true);
    }
  } catch (Exception e) {
    System.out.println(e.toString());
  }
}

/*
 * Parses lookup tables
 */
private void parseLookUpTable(Module module) throws IOException,
    MaaohaException {
  String name;
  String type = "";
  ArrayList<Integer> elements = new ArrayList<Integer>();

  // rading name
  String token = getNextToken(false);
  name = token;
  token = getNextToken(false);

  // rading type
  if (!token.equalsIgnoreCase(":"))
    throw new MaaohaException(
        "GezelParser->parseLookUpTable::Expected character ':'");
  token = getNextToken(false);
  while (!token.equalsIgnoreCase("=")) {
    type += token;
    token = getNextToken(false);
  }
  type.replaceAll(" ", "");

  // reading values
  token = getNextToken(false);
  if (!token.equalsIgnoreCase("{"))
    throw new MaaohaException(
        "GezelParser->parseLookUpTable::Expected character '{'");

  while (!token.equalsIgnoreCase("}")) {
    token = getNextToken(false);
    int value = Integer.parseInt(token);
    elements.add(value);

    token = getNextToken(false);
    if (!token.equals(",") && !token.equals("}"))
      throw new MaaohaException(
          "GezelParser->parseLookUpTable::Expected character ',' or '}'");
  }

  token = getNextToken(false);
  if (!token.equals(";"))
    throw new MaaohaException(
        "GezelParser->parseLookUpTable::Expected character ';'");

  // adding new lookup table to the datapath
  module.getDatapath().addNewLookUpTable(name, type, elements);
}

/*
 * Parses sfg
 */
private void parseSfg(Module module) throws IOException, MaaohaException {
  String token = getNextToken(false);
  Sfg sfg = new Sfg(token);
```

```
  parseSfgExpressions(sfg, module);
}

/*
 * Parses expression
 */
private void parseSfgExpressions(Sfg sfg, Module module)
    throws IOException, MaaohaException {
  String token = getNextToken(false);
  boolean addExpression = true;
  // loop until the end of sfg
  while (!token.equalsIgnoreCase("}")) {
    addExpression = true;
    String strExpression = "";
    token = getNextToken(false);

    if (!token.equalsIgnoreCase("}")) {
      boolean leftSide = true;
      DatapathElement leftSideElement = null;
      ArrayList<DatapathElement> rightSideElements =
         new ArrayList<DatapathElement>();

      while (!token.equals(";")) {
        // ignore display directive while parsing
        if (token.equals("$display")) {
          addExpression = false;
          int bracketsNo = 1;
          token = getNextToken(false);
          if (!token.equals("("))
            throw new MaaohaException(
                "GezelParser->parseSfgExpressions::Expected '('");

          while (bracketsNo > 0) {
            token = getNextToken(false);
            if (token.equals("("))
              bracketsNo++;
            else if (token.equals(")"))
              bracketsNo--;
            if (token == null)
              throw new MaaohaException(
                  "GezelParser->parseSfgExpressions::
                       Unclosed $display procedure");
          }
          token = getNextToken(false);
        } else {
          // parse single expression
          if (token.equals("="))
            leftSide = false;

          for (int i = 0; i < module.getDatapath()
              .getElementsNo(); i++) {
            if (module.getDatapath().getElement(i).getName()
                .equals(token)) {
              DatapathElement de = module.getDatapath()
                  .getElement(i);
              if (leftSide)
                leftSideElement = de;
              else {
                if (!rightSideElements.contains(de))
                  rightSideElements.add(de);
              }

              i = module.getDatapath().getElementsNo();
            }
          }
          strExpression += token;
          token = getNextToken(false);
        }
```

```
      }

      if (addExpression)
        sfg.addExpression(new Expression(strExpression,
            leftSideElement, rightSideElements, module
                .getDatapath().getLookupTables()));
    }
  }
  module.getDatapath().addSfg(sfg);
}

/*
 * Parses transition condition
 */
private String parseTransCondition() throws IOException {
  String condition = "";
  String token = getNextToken(false); // "("

  if (!token.equalsIgnoreCase("("))
    throw new IOException("Gezel file error line = " + lineNo);

  int brackets = 1;
  while (brackets > 0) {
    token = getNextToken(false); //
    if (token.equalsIgnoreCase("("))
      brackets++;
    if (token.equalsIgnoreCase(")"))
      brackets--;
    if (brackets > 0) {
      condition += token;
    }
  }

  return condition;
}

/*
 * Parses transition
 */
private void parseTransition(Module module, Fsm fsm, String state)
    throws IOException, MaaohaException {
  String token = getNextToken(false); // "if" || "("

  if (token.equals("if")) {
    String condition = parseTransCondition();
    token = getNextToken(false); // "then"
    String strSfgs = parseTransSfg(true);
    token = getNextToken(false); // next state name
    String nextState = token;
    token = getNextToken(false); // ";"

    module.addFsmTransitionFromString(state, condition, strSfgs,
        nextState);

    boolean notEnd = true;

    while (notEnd) {
      token = getNextToken(false); // "else"
      token = getNextToken(false); // "if" || "("
      if (token.equalsIgnoreCase("("))
        notEnd = false;
      else if (token.equalsIgnoreCase("if")) {
        condition = parseTransCondition();
        token = getNextToken(false); // "then"
        strSfgs = parseTransSfg(true);

        token = getNextToken(false); // next state name
        nextState = token;
```

```
        token = getNextToken(false); // ";"

        module.addFsmTransitionFromString(state, condition,
            strSfgs, nextState);
      }
    }

    strSfgs = parseTransSfg(true);

    token = getNextToken(false); // next state name
    nextState = token;
    token = getNextToken(false); // ";"

    module.addFsmTransitionFromString(state, "1", strSfgs, nextState);
  } else if (token.equalsIgnoreCase("(")) {
    String strSfgs = parseTransSfg(true);

    token = getNextToken(false); // next state name
    String nextState = token;
    token = getNextToken(false); // ";"
    module.addFsmTransitionFromString(state, "1", strSfgs, nextState);
  } else
    throw new IOException("Gezel file error line = " + lineNo);
}

/*
 * Parses sfgs executed by transition
 */
private String parseTransSfg(boolean withFirstBracket) throws IOException {
  String token = "";
  String transition = "";

  if (withFirstBracket) {
    token = getNextToken(false); // "("
    transition += "(" + token;
  }

  while (!token.equalsIgnoreCase(")")) {
    token = getNextToken(false); // sfg name
    transition += token;
  }
  token = getNextToken(false); // "->"

  return transition;
}

/*
 * Parses use of sub-module
 */
private void parseUseModule(Module module, Model model) throws IOException,
    MaaohaException {
  String token = getNextToken(false);
  String name = token;
  ArrayList<DatapathElement> parameters = new ArrayList<DatapathElement>();

  token = getNextToken(false);
  if (token.equalsIgnoreCase("(")) {
    while (!token.equalsIgnoreCase(")")) {
      token = getNextToken(false);
      for (int i = 0; i < module.getDatapath().getElementsNo(); i++) {
        if (module.getDatapath().getElement(i).getName().equals(
            token)) {
          parameters.add(module.getDatapath().getElement(i));
          i = module.getDatapath().getElementsNo();
        }
      }
      token = getNextToken(false);
    }
```

```
    }

    Module moduleType = model.getModule(name);
    if (moduleType == null)
      moduleType = findModuleCopy(name, moduleCopies);
    if (moduleType == null)
      throw new MaaohaException(
          "GezelParser->parseUseModule::Cannot fing module: " + name);
    UseModule um = new UseModule(name, moduleType, parameters);
    module.getDatapath().addUseModule(um);
  }

  /*
   * Reads and prepares for parsing next line of GEZEL program
   */
  private boolean readNewLine(boolean possibleEnd) throws IOException {
    while ((line == null && tokenizer == null)
        || (tokenizer != null && !tokenizer.hasMoreElements())) {
      // read new line
      line = in.readLine();
      lineNo++;

      // no more lines
      if (line == null) {
        if (possibleEnd)
          return true;
        else
          throw new IOException("Gezel file error line = " + lineNo);
      }

      line = MaaohaTools.separateTokens(line);
      tokenizer = new StringTokenizer(line);
    }
    return false;
  }
}
```

B.4.3 NuSMVGenerator.java

```
package maaoha.source;

import java.util.ArrayList;

import maaoha.MaaohaException;
import maaoha.model.Condition;
import maaoha.model.Datapath;
import maaoha.model.DatapathElement;
import maaoha.model.Fsm;
import maaoha.model.FsmState;
import maaoha.model.FsmTransition;
import maaoha.model.Model;
import maaoha.model.Module;
import maaoha.model.Sfg;
import maaoha.model.UseModule;
import maaoha.model.expression.BasicExpression;
import maaoha.model.expression.BinaryOperator;
import maaoha.model.expression.BitSelection;
import maaoha.model.expression.Constant;
import maaoha.model.expression.LookupTableOperator;
import maaoha.model.expression.TernaryOperator;
import maaoha.model.expression.UnaryOperator;
import maaoha.model.expression.Variable;
import maaoha.tools.MaaohaTools;

/*
 * Generates NuSMV program
 */
```

```
public class NuSMVGenerator {
  /*
   * Auxiliary structure for storage of expression cases
   */
  static class ExpressionCases {
    public BasicExpression exp1;
    public BasicExpression exp2;
    public BasicExpression cond;
  }

  /*
   * Auxiliary structure for storage of conditional cases
   */
  static class ExpressionConditionCases {
    public ArrayList<BasicExpression> eList;
    public ArrayList<BasicExpression> cList;
  }

  static public String eol = "\n";
  static public String ctl = "ctl_";
  static public String mod = "mod_";
  static public String st = "st_";
  static public String to = "_to_";
  static public String sig = "sig_";
  static public String reg = "reg_";
  static public String inp = "inp_";
  static public String out = "out_";
  static public String sfg = "sfg_";
  static public String um = "um_";
  static public String init = "INIT";
  static public String trans = "TRANS";
  static public String cs = "case";
  static public String sc = "esac";

  static public String sp = "    ";

  static public String sysmod = "system";

  /*
   * Converts basic expressions to NuSMV
   */
  static String convertBasicExpressionToNuSMV(BasicExpression exp)
      throws MaaohaException {
    int expType = exp.getType();

    if (expType == BasicExpression.Brackets) {
      return ("("
          + convertBasicExpressionToNuSMV(((UnaryOperator) exp)
              .getExpression()) + ")");
    } else if (expType == BasicExpression.Negation) {
      return ("!" + convertBasicExpressionToNuSMV(((UnaryOperator) exp)
          .getExpression()));
    } else if (expType == BasicExpression.TernaryConditional) {
      TernaryOperator top = (TernaryOperator) exp;
      return (convertBasicExpressionToNuSMV(top.getExpression1()) + " ? "
          + convertBasicExpressionToNuSMV(top.getExpression2())
          + " : " + convertBasicExpressionToNuSMV(top
          .getExpression3()));
    } else if (expType == BasicExpression.ShiftLeft
        || expType == BasicExpression.ShiftRight) {
      String operatorSign = "";
      if (expType == BasicExpression.ShiftLeft)
        operatorSign = " << ";
      else if (expType == BasicExpression.ShiftRight)
        operatorSign = " >> ";

      BinaryOperator bop = (BinaryOperator) exp;
      int vt1 = bop.getExpression1().getValueType();
```

```
  int vt2 = bop.getExpression2().getValueType();

  int vt1up = 1;
  int vt1up0 = 0;
  while (vt1up < vt1) {
    vt1up *= 2;
    vt1up0++;
  }

  if (vt1up > vt1) {
    if (vt1up0 > vt2)
      return ("((0d"
          + (vt1up - vt1)
          + "_0 :: ("
          + convertBasicExpressionToNuSMV(bop
              .getExpression1())
          + "))"
          + operatorSign
          + "(("
          + convertBasicExpressionToNuSMV(bop
              .getExpression2()) + ")))[" + (vt1 - 1) + ":0]");
    return ("((0d" + (vt1up - vt1) + "_0 :: ("
        + convertBasicExpressionToNuSMV(bop.getExpression1())
        + "))" + operatorSign + "(("
        + convertBasicExpressionToNuSMV(bop.getExpression2())
        + ")[" + (vt1up0 - 1) + ":0]))[" + (vt1 - 1) + ":0]");
  } else {
    if (vt1up0 > vt2)
      return ("(((("
          + convertBasicExpressionToNuSMV(bop
              .getExpression1())
          + "))"
          + operatorSign
          + "(("
          + convertBasicExpressionToNuSMV(bop
              .getExpression2()) + ")))[" + (vt1 - 1) + ":0]");
    return ("(((("
        + convertBasicExpressionToNuSMV(bop.getExpression1())
        + "))" + operatorSign + "(("
        + convertBasicExpressionToNuSMV(bop.getExpression2())
        + ")[" + (vt1up0 - 1) + ":0]))[" + (vt1 - 1) + ":0]");
  }
} else if (expType == BasicExpression.BitwiseAnd
    || expType == BasicExpression.BitwiseOr
    || expType == BasicExpression.BitwiseXor
    || expType == BasicExpression.Addition
    || expType == BasicExpression.Subtraction) {
  String operatorSign = "";
  if (expType == BasicExpression.BitwiseAnd)
    operatorSign = " & ";
  else if (expType == BasicExpression.BitwiseOr)
    operatorSign = " | ";
  else if (expType == BasicExpression.BitwiseXor)
    operatorSign = " xor ";

  else if (expType == BasicExpression.Addition)
    operatorSign = " + ";
  else if (expType == BasicExpression.Subtraction)
    operatorSign = " - ";

  BinaryOperator bop = (BinaryOperator) exp;
  int vt1 = bop.getExpression1().getValueType();
  int vt2 = bop.getExpression2().getValueType();

  System.out.println("types : " + vt1 + " , " + vt2);

  if (vt1 > vt2)
    return (convertBasicExpressionToNuSMV(bop.getExpression1())
```

```
        + operatorSign + "(( 0d" + (vt1 - vt2) + "_0 ) :: "
        + convertBasicExpressionToNuSMV(bop.getExpression2()) + ")");
  else if (vt2 > vt2)
    return ("(( 0d" + (vt1 - vt2) + "_0 ) :: "
        + convertBasicExpressionToNuSMV(bop.getExpression1())
        + ")" + operatorSign
        + convertBasicExpressionToNuSMV(bop.getExpression2()) + ")");

  return (convertBasicExpressionToNuSMV(bop.getExpression1())
      + operatorSign + convertBasicExpressionToNuSMV(bop
      .getExpression2()));
} else if (expType == BasicExpression.Concatenation) {
  BinaryOperator bop = (BinaryOperator) exp;
  return (convertBasicExpressionToNuSMV(bop.getExpression1())
      + " :: " + convertBasicExpressionToNuSMV(bop
      .getExpression2()));
} else if (expType == BasicExpression.Equality
    || expType == BasicExpression.LessOrEqual
    || expType == BasicExpression.GreaterOrEqual
    || expType == BasicExpression.Less
    || expType == BasicExpression.Greater) {
  String operatorSign = "";
  if (expType == BasicExpression.Equality)
    operatorSign = " = ";
  else if (expType == BasicExpression.LessOrEqual)
    operatorSign = " <= ";
  else if (expType == BasicExpression.GreaterOrEqual)
    operatorSign = " >= ";
  else if (expType == BasicExpression.Less)
    operatorSign = " < ";
  else if (expType == BasicExpression.Greater)
    operatorSign = " > ";

  BinaryOperator bop = (BinaryOperator) exp;
  int vt1 = bop.getExpression1().getValueType();
  int vt2 = bop.getExpression2().getValueType();

  if (vt1 > vt2)
    return ("word1("
        + convertBasicExpressionToNuSMV(bop.getExpression1())
        + operatorSign + "(( 0d" + (vt1 - vt2) + "_0 ) :: "
        + convertBasicExpressionToNuSMV(bop.getExpression2()) + "))");
  else if (vt2 > vt2)
    return ("word1((( 0d" + (vt1 - vt2) + "_0 ) :: "
        + convertBasicExpressionToNuSMV(bop.getExpression1())
        + ")" + operatorSign
        + convertBasicExpressionToNuSMV(bop.getExpression2()) + "))");

  return ("word1("
      + convertBasicExpressionToNuSMV(bop.getExpression1())
      + operatorSign
      + convertBasicExpressionToNuSMV(bop.getExpression2()) + ")");
} else if (expType == BasicExpression.ConstantValue) {
  Constant constant = (Constant) exp;
  String strconstant = "";

  strconstant = convertValueToNuSMV(constant.getValue());
  return strconstant;
} else if (expType == BasicExpression.DatapathElement) {
  DatapathElement de = ((Variable) exp).getElement();
  String strname = "";
  if (de.getElementType().equals(Model.typeInput))
    strname = "next(" + inp + de.getName() + ")";
  else if (de.getElementType().equals(Model.typeOutput)) {
    throw new MaaohaException(
        "NuSMVGenerator->convertBasicExpressionToNuSMV::
            Output on the right side of expression");
  } else if (de.getElementType().equals(Model.typeSignal))
```

```
      strname = "next(" + sig + de.getName() + ")";
    else if (de.getElementType().equals(Model.typeRegister))
      strname = reg + de.getName();
    else
      strname = "ERROR unknown variable type";

    return strname;
  } else if (expType == BasicExpression.BitSelection) {
    DatapathElement de = ((BitSelection) exp).getElement();
    String str = "";
    if (de.getElementType().equals(Model.typeInput))
      str = inp + de.getName();
    else if (de.getElementType().equals(Model.typeOutput))
      str = out + de.getName();
    else if (de.getElementType().equals(Model.typeSignal))
      str = sig + de.getName();
    else if (de.getElementType().equals(Model.typeRegister))
      str = reg + de.getName();
    else
      return "ERROR unknown variable type";

    BitSelection bitsel = (BitSelection) exp;

    str += "[" + bitsel.getFirstBit() + ":" + bitsel.getLastBit() + "]";

    return str;
  }
  return "ERROR";
}

/*
 * Converts condition to NuSMV
 */
static private String convertConditionToNuSMV(BasicExpression exp)
    throws MaaohaException {
  String str = "(" + convertBasicExpressionToNuSMV(exp) + ") > 0d"
      + exp.getValueType() + "_0";
  return str;
}

/*
 * Converts condition to NuSMV
 */
static private String convertConditionToNuSMV(Condition cond)
    throws MaaohaException {
  String str = "("
      + convertBasicExpressionToNuSMV(cond.getBasicExpression())
      + ") > 0d" + cond.getBasicExpression().getValueType() + "_0";
  return str;
}

/*
 * Converts expression to NuSMV
 */
static private String convertExpressionToNuSMV(BasicExpression bex, int type)
    throws MaaohaException {
  int type2 = bex.getValueType();
  String str = "";
  if (type == type2)
    str = convertBasicExpressionToNuSMV(bex);
  else if (type > type2)
    str = "0d" + (type - type2) + "_0 :: ("
        + convertBasicExpressionToNuSMV(bex) + ")";
  else if (type < type2)
    str = "(" + convertBasicExpressionToNuSMV(bex) + ")["
        + (type2 - type) + ":0]";

  return str;
```

```
}

/*
 * Converts variable type to NuSMV
 */
static private String convertTypeToNuSMV(String str) {
  int charPos = str.indexOf("ns(");
  int charPos2 = str.indexOf(")");
  if (charPos == -1 || charPos2 == -1)
    return "ERROR: cant parse type";
  int number = Integer.parseInt(str.substring(charPos + 3, charPos2));

  return "word[" + number + "]";
}

/*
 * Converts value to NuSMV
 */
static private String convertValueToNuSMV(int value) {
  int number = 0;
  int val = value;
  while (val > 0) {
    number++;
    val /= 2;
  }
  if (number == 0)
    number = 1;
  return "0d" + number + "_" + value;
}

/*
 * Converts value to NuSMV for initialization of variables
 */
static private String convertValueToNuSMVforInit(String type, int value) {
  int charPos = type.indexOf("ns(");
  int charPos2 = type.indexOf(")");
  if (charPos == -1 || charPos2 == -1)
    return "ERROR: cant parse type";
  int number = Integer.parseInt(type.substring(charPos + 3, charPos2));

  return "0d" + number + "_" + value;
}

/*
 * Divide simple cases of expressions according to lookup table cases
 */
static protected void divideCasesToLookupTableCases(BasicExpression be,
    ExpressionConditionCases newEcc) throws MaaohaException {
  newEcc.eList = new ArrayList<BasicExpression>();
  newEcc.cList = new ArrayList<BasicExpression>();

  newEcc.eList.add(be);
  newEcc.cList.add(null);

  boolean gogogo = true;

  while (gogogo) {
    gogogo = false;

    ExpressionConditionCases sumList = new ExpressionConditionCases();
    sumList.eList = new ArrayList<BasicExpression>();
    sumList.cList = new ArrayList<BasicExpression>();

    for (int i = 0; i < newEcc.eList.size(); i++) {

      ExpressionConditionCases oneList = new ExpressionConditionCases();
      oneList.eList = new ArrayList<BasicExpression>();
      oneList.cList = new ArrayList<BasicExpression>();
```

```
      if (newEcc.eList.get(i).containsExpressionOfType(
          BasicExpression.LookupOperator)) {
        gogogo = true;
        divideCasesToLookupTableCasesStep(newEcc.eList.get(i),
            oneList);
      }

      sumList.eList.addAll(oneList.eList);
      sumList.cList.addAll(oneList.cList);
    }
    if (gogogo) {
      newEcc.eList = sumList.eList;
      newEcc.cList = sumList.cList;
    }
  }
}

/*
 * Divide simple cases of expressions according to lookup table cases -
 * single step
 */
static protected boolean divideCasesToLookupTableCasesStep(
    BasicExpression be, ExpressionConditionCases ecc)
    throws MaaohaException {
  if (be.getType() == BasicExpression.LookupOperator) {
    BinaryOperator bop = (BinaryOperator) be;
    LookupTableOperator ltop = (LookupTableOperator) bop
        .getExpression1();
    int casesNo = ltop.getTable().getElementsNo();
    for (int i = 0; i < casesNo; i++) {
      ecc.eList.add(new Constant(ltop.getTable().getElement(i)));
      ecc.cList.add(new BinaryOperator(BasicExpression.Equality, bop
          .getExpression2().clone(), new Constant(i)));
    }
    return true;
  }
  boolean end = false;
  if (be instanceof BinaryOperator) {
    BinaryOperator bop = (BinaryOperator) be;
    end = divideCasesToLookupTableCasesStep(bop.getExpression1(), ecc);
    if (end) {
      ArrayList<BasicExpression> eList2 = new ArrayList<BasicExpression>();

      for (int i = 0; i < ecc.eList.size(); i++) {
        BasicExpression bex = ecc.eList.get(i);
        BinaryOperator thisnode = new BinaryOperator(bop.getType(),
            bex, bop.getExpression2().clone());

        eList2.add(thisnode);
      }
      ecc.eList = eList2;

      return true;
    }
    end = divideCasesToLookupTableCasesStep(bop.getExpression2(), ecc);
    if (end) {
      ArrayList<BasicExpression> eList2 = new ArrayList<BasicExpression>();

      for (int i = 0; i < ecc.eList.size(); i++) {
        BasicExpression bex = ecc.eList.get(i);
        BinaryOperator thisnode = new BinaryOperator(bop.getType(),
            bop.getExpression2().clone(), bex);

        eList2.add(thisnode);
      }
      ecc.eList = eList2;
```

```
      return true;
    }
    throw new MaaohaException(
        "NuSMVGeneratot->divideCasesToLookupTableCasesStep::No true returned");
  }
  if (be instanceof UnaryOperator) {
    UnaryOperator uop = (UnaryOperator) be;
    end = divideCasesToLookupTableCasesStep(uop.getExpression(), ecc);
    if (end) {
      ArrayList<BasicExpression> eList2 = new ArrayList<BasicExpression>();

      for (int i = 0; i < ecc.eList.size(); i++) {
        BasicExpression bex = ecc.eList.get(i);
        UnaryOperator thisnode = new UnaryOperator(uop.getType(),
            bex);

        eList2.add(thisnode);
      }
      ecc.eList = eList2;

      return true;
    }
    throw new MaaohaException(
        "NuSMVGeneratot->divideCasesToLookupTableCasesStep::No true returned");
  } else
    return false;
}

/*
 * Divides expression into cases
 */
protected static void divideExpressionToSimpleCases(BasicExpression be,
    ExpressionConditionCases ecc) throws MaaohaException {
  ArrayList<BasicExpression> exCases = new ArrayList<BasicExpression>();
  ArrayList<BasicExpression> coCases = new ArrayList<BasicExpression>();

  BasicExpression be2 = be.clone();
  exCases.add(be2);
  coCases.add(null);

  int iterationNo;
  boolean gogogo = true;

  while (gogogo) {

    gogogo = false;
    if (exCases.size() != coCases.size())
      throw new MaaohaException(
          "NuSMVGenerator->divideExpressionToSimpleCases:
              Number of cases and caondition do not match");
    else
      iterationNo = exCases.size();

    for (int i = 0; i < iterationNo; i++) {
      BasicExpression bex = exCases.get(i);
      if (bex
          .getNoOfExpressionsOfType(BasicExpression.TernaryConditional) > 0) {
        gogogo = true;
        ExpressionCases ec = new ExpressionCases();
        divideExpressionToSimpleCasesStep(bex, ec);
        exCases.add(ec.exp1);
        exCases.add(ec.exp2);
        coCases.add(ec.cond);
        coCases.add(null);
        exCases.remove(i);
        exCases.add(i, null);

      }
```

```
    }
    for (int i = 0; i < exCases.size(); i++) {
      if (exCases.get(i) == null) {
        exCases.remove(i);
        coCases.remove(i);
        i--;
      }
    }
  }
  ecc.eList = exCases;
  ecc.cList = coCases;
}

/*
 * Single step of expression to simple case division
 */
static protected boolean divideExpressionToSimpleCasesStep(
    BasicExpression be, ExpressionCases ec) throws MaaohaException {
  if (be.getType() == BasicExpression.TernaryConditional) {
    TernaryOperator top = (TernaryOperator) be;
    ec.exp1 = top.getExpression2().clone();
    ec.exp2 = top.getExpression3().clone();
    ec.cond = top.getExpression1().clone();
    return true;
  }
  boolean end = false;
  if (be instanceof BinaryOperator) {
    BinaryOperator bop = (BinaryOperator) be;
    end = divideExpressionToSimpleCasesStep(bop.getExpression1(), ec);
    if (end) {
      BinaryOperator thisnode1 = new BinaryOperator(bop.getType(),
          ec.exp1, bop.getExpression2().clone());
      BinaryOperator thisnode2 = new BinaryOperator(bop.getType(),
          ec.exp2, bop.getExpression2().clone());
      ec.exp1 = thisnode1;
      ec.exp2 = thisnode2;
      return true;
    }
    end = divideExpressionToSimpleCasesStep(bop.getExpression2(), ec);
    if (end) {
      BinaryOperator thisnode1 = new BinaryOperator(bop.getType(),
          bop.getExpression1().clone(), ec.exp1);
      BinaryOperator thisnode2 = new BinaryOperator(bop.getType(),
          bop.getExpression1().clone(), ec.exp2);
      ec.exp1 = thisnode1;
      ec.exp2 = thisnode2;
      return true;
    }
    throw new MaaohaException(
        "NuSMVGeneratot->divideExpressionToSimpleCasesStep::No true returned");
  }
  if (be instanceof UnaryOperator) {
    UnaryOperator uop = (UnaryOperator) be;
    end = divideExpressionToSimpleCasesStep(uop.getExpression(), ec);
    if (end) {
      UnaryOperator thisnode1 = new UnaryOperator(uop.getType(),
          ec.exp1);
      UnaryOperator thisnode2 = new UnaryOperator(uop.getType(),
          ec.exp2);
      ec.exp1 = thisnode1;
      ec.exp2 = thisnode2;
      return true;
    }
    throw new MaaohaException(
        "NuSMVGeneratot->divideExpressionToSimpleCasesStep::No true returned");
  } else
    return false;
}
```

```
/*
 * static class ConditionAndExpression { public
 * ConditionAndExpression(String condition) { this.condition = condition; }
 *
 * String condition; }
 */

/*
 * Generates NuSMV program for specified model
 */
static public String GenerateNuSMVModel(Model model) throws MaaohaException {
  String program = "";
  for (int i = 0; i < model.getModulesNo(); i++) {
    Module module = model.getModule(i);
    Fsm fsm = module.getFsm();
    Datapath datapath = module.getDatapath();
    String m_program = "";

    // generate module name
    if (module.getName().equals(sysmod))
      m_program += "MODULE " + "main";
    else
      m_program += "MODULE " + mod + module.getName();

    // generate module parameters
    if (module.getDatapath().hasInputsOutputs()) {
      boolean firstIO = true;
      m_program += "(";
      for (int i1 = 0; i1 < datapath.getElementsNo(); i1++) {
        if (datapath.getElement(i1).getElementType().equals(
            Model.typeInput)) {
          if (!firstIO)
            m_program += ", ";
          m_program += inp + datapath.getElement(i1).getName();
          firstIO = false;
        } else if (datapath.getElement(i1).getElementType().equals(
            Model.typeOutput)) {
          if (!firstIO)
            m_program += ", ";
          m_program += out + datapath.getElement(i1).getName();
          firstIO = false;
        }
      }
      m_program += ")";
    }

    m_program += eol + eol + "VAR" + eol + eol;

    if (fsm.getStatesNo() > 0) {
      // generate controller declaratiion
      m_program += ctl + module.getName() + " : " + " {";

      for (int i1 = 0; i1 < fsm.getStatesNo(); i1++) {
        FsmState state = fsm.getFsmState(i1);
        // add gezel states
        if (i1 > 0)
          m_program += ", ";
        m_program += st + state.getName();
      }
      m_program += "};" + eol + eol;
    }

    // generate datapath elements declaration
    for (int i1 = 0; i1 < datapath.getElementsNo(); i1++) {
      DatapathElement de = datapath.getElement(i1);
      if (de.getElementType().equals(Model.typeRegister)
          || de.getElementType().equals(Model.typeSignal)) {
```

```
    if (de.getElementType().equals(Model.typeRegister))
      m_program += reg;
    else if (de.getElementType().equals(Model.typeSignal))
      m_program += sig;
    else if (de.getElementType().equals(Model.typeOutput))
      m_program += out;
    m_program += de.getName() + " : "
        + convertTypeToNuSMV(de.getValueType()) + ";" + eol;
  }
}
m_program += eol;

// use modules
for (int i1 = 0; i1 < datapath.getUseModulesNo(); i1++) {
  UseModule umod = datapath.getUseModule(i1);
  m_program += um + umod.getName() + " : " + mod
      + umod.getModule().getName() + "(";
  for (int i2 = 0; i2 < umod.getParametersNo(); i2++) {
    if (i2 > 0)
      m_program += ", ";
    if (umod.getParameter(i2).getElementType().equals(
        Model.typeRegister))
      m_program += reg + umod.getParameter(i2).getName();
    else if (umod.getParameter(i2).getElementType().equals(
        Model.typeSignal))
      m_program += sig + umod.getParameter(i2).getName();
  }
  m_program += ");" + eol;
}
m_program += eol;

// signal flow graphs declaration
for (int i1 = 0; i1 < datapath.getSfgsNo(); i1++) {
  m_program += sfg + datapath.getSfg(i1).getName()
      + " : boolean;" + eol;
}

if (fsm.getStatesNo() > 0) {
  if (fsm.getInitialState() == null)
    fsm.setInitialState(fsm.getFsmState(0));

  // generate controller definition
  m_program += eol + init + eol + ctl + module.getName() + " = "
      + st + fsm.getInitialState().getName() + ";" + eol
      + trans + eol + "next(" + ctl + module.getName() + ")"
      + " = " + eol + sp + cs + eol;

  for (int i1 = 0; i1 < fsm.getStatesNo(); i1++) {
    FsmState state = fsm.getFsmState(i1);
    for (int i2 = 0; i2 < state.getTransitionsNo(); i2++) {
      FsmTransition transition = state.getTransistion(i2);
      m_program += sp
          + sp
          + ctl
          + module.getName()
          + " = "
          + st
          + state.getName()
          + " & "
          + convertConditionToNuSMV(transition
              .getCondition()) + " : " + st
          + transition.getNextState().getName() + ";"
          + eol;
    }
  }
  m_program += sp + sp + "1 : " + ctl + module.getName() + ";"
      + eol + sp + sc + eol;
}
```

```
// generate elements definition
for (int i1 = 0; i1 < datapath.getElementsNo(); i1++) {
  DatapathElement de = datapath.getElement(i1);
  if (de.getElementType().equals(Model.typeRegister)
      || (de.getElementType().equals(Model.typeSignal) && !(datapath
          .isElementUsedAsOutputForModule(de)))
      || de.getElementType().equals(Model.typeOutput)) {
    if (de.getElementType().equals(Model.typeRegister)) {
      m_program += eol
          + init
          + eol
          + reg
          + de.getName()
          + " = "
          + convertValueToNuSMVforInit(de.getValueType(),
              0) + ";" + eol + trans + eol + "next("
          + reg + de.getName() + ")" + " = " + eol + sp
          + cs + eol;

      for (int i2 = 0; i2 < datapath.getSfgsNo(); i2++) {
        Sfg sfgraph = datapath.getSfg(i2);
        m_program = writeSingleLineOfExpression(m_program,
            sfgraph, de);
      }
      m_program += sp + sp + "1 : " + reg + de.getName()
          + ";" + eol + sp + sc + eol;
    } else if (de.getElementType().equals(Model.typeSignal)) {
      m_program += eol
          + init
          + eol
          + sig
          + de.getName()
          + " = "
          + convertValueToNuSMVforInit(de.getValueType(),
              0) + ";" + eol + trans + eol + "next("
          + sig + de.getName() + ")" + " = " + eol + sp
          + cs + eol;

      for (int i2 = 0; i2 < datapath.getSfgsNo(); i2++) {
        Sfg sfgraph = datapath.getSfg(i2);
        m_program = writeSingleLineOfExpression(m_program,
            sfgraph, de);
      }
      m_program += sp + sp + "1 : " + sig + de.getName()
          + ";" + eol + sp + sc + eol;
    } else if (de.getElementType().equals(Model.typeOutput)) {
      m_program += eol
          + init
          + eol
          + out
          + de.getName()
          + " = "
          + convertValueToNuSMVforInit(de.getValueType(),
              0) + ";" + eol;
      m_program += trans + eol + "next(" + out + de.getName()
          + ")" + " = " + eol + sp + cs + eol;

      for (int i2 = 0; i2 < datapath.getSfgsNo(); i2++) {
        Sfg sfgraph = datapath.getSfg(i2);
        m_program = writeSingleLineOfExpression(m_program,
            sfgraph, de);
      }
      m_program += sp + sp + "1 : " + out + de.getName()
          + ";" + eol + sp + sc + eol;
    }
  }
}
```

```
    // generate sfg definition
    for (int i1 = 0; i1 < datapath.getSfgsNo(); i1++) {
      Sfg sfgraph = datapath.getSfg(i1);

      m_program += eol + init + eol + sfg + sfgraph.getName() + " = "
          + "FALSE" + ";" + eol + trans + eol + "next(" + sfg
          + sfgraph.getName() + ")" + " = " + eol + sp + cs + eol;

      for (int i2 = 0; i2 < fsm.getStatesNo(); i2++) {
        FsmState state = fsm.getFsmState(i2);
        for (int i3 = 0; i3 < state.getTransitionsNo(); i3++) {
          FsmTransition transition = state.getTransistion(i3);
          if (transition.containsSfg(sfgraph)) {
            m_program += sp
                + sp
                + ctl
                + module.getName()
                + " = "
                + st
                + state.getName()
                + " & "
                + convertConditionToNuSMV(transition
                    .getCondition()) + " : TRUE;" + eol;
          } else {
            m_program += sp
                + sp
                + ctl
                + module.getName()
                + " = "
                + st
                + state.getName()
                + " & "
                + convertConditionToNuSMV(transition
                    .getCondition()) + " : FALSE;"
                + eol;
          }
        }
      }
      m_program += sp + sp + "1 : FALSE;" + eol + sp + sc + eol;

    }
    program += m_program;
  }
  return program;
}

/*
 * Generates single line of expression
 */
static protected String writeSingleLineOfExpression(String m_program,
    Sfg sfgraph, DatapathElement de) throws MaaohaException {
  int expressionIndex = sfgraph.doesCalculate(de);
  if (expressionIndex > -1) {
    ExpressionConditionCases ecc = new ExpressionConditionCases();
    divideExpressionToSimpleCases(sfgraph
        .getExpression(expressionIndex).getBasicExpression(), ecc);
    int valueType = MaaohaTools.extractIntFromType(sfgraph
        .getExpression(expressionIndex).getLeftSideElement()
        .getValueType());
    ExpressionConditionCases sumEcc = new ExpressionConditionCases();
    sumEcc.eList = new ArrayList<BasicExpression>();
    sumEcc.cList = new ArrayList<BasicExpression>();
    for (int i = 0; i < ecc.eList.size(); i++) {
      ExpressionConditionCases newEcc = new ExpressionConditionCases();
      divideCasesToLookupTableCases(ecc.eList.get(i), newEcc);
      for (int j = 0; j < newEcc.cList.size(); j++) {
        if ((ecc.cList.get(i) == null)
```

```
              && (newEcc.cList.get(j) == null))
            sumEcc.cList.add(new Constant(1));
          else if (ecc.cList.get(i) == null)
            sumEcc.cList.add(newEcc.cList.get(j));
          else if (newEcc.cList.get(j) == null)
            sumEcc.cList.add(ecc.cList.get(i));
          else {
            BinaryOperator bop = new BinaryOperator(
                BasicExpression.BitwiseAnd, ecc.cList.get(i),
                newEcc.cList.get(j));
            sumEcc.cList.add(bop);
          }
        }
        sumEcc.eList.addAll(newEcc.eList);
      }

      for (int i = 0; i < sumEcc.eList.size(); i++)
        m_program += sp
            + sp
            + "next("
            + sfg
            + sfgraph.getName()
            + ") = TRUE & "
            + convertConditionToNuSMV(sumEcc.cList.get(i))
            + " : "
            + convertExpressionToNuSMV(sumEcc.eList.get(i),
                valueType) + ";" + eol;

    }
    return m_program;
  }
}
```

B.5 Package maaoha.tools

B.5.1 MaaohaTools.java

```
package maaoha.tools;

/*
 * Class provides auxiliary functions for string manipulation
 */
public class MaaohaTools {
  /*
   * Removes left side of the expression
   */
  static public String cutLeftSide(String expression) {
    expression = expression.substring(expression.indexOf("=", 0) + 1);
    return expression;
  }

  /*
   * Extracts number of bits from type
   */
  static public int extractIntFromType(String type) {
    int charPos = type.indexOf("ns(");
    int charPos2 = type.indexOf(")");
    if (charPos == -1 || charPos2 == -1)
      return -1;
    int number = Integer.parseInt(type.substring(charPos + 3, charPos2));

    return number;
  }

  /*
   * Separates tokens
```

```
 */
static public String separateTokens(String expression) {
  // separate tokens
  expression = expression.replaceAll("=", " = ");
  expression = expression.replaceAll("=  =", " == ");
  expression = expression.replaceAll("! =", " != ");
  expression = expression.replaceAll("//", " // ");
  expression = expression.replaceAll("~", " ~ ");
  expression = expression.replaceAll("\\^", " ^ ");
  expression = expression.replaceAll(";", " ; ");
  expression = expression.replaceAll(":", " : ");
  expression = expression.replaceAll("&", " & ");
  expression = expression.replaceAll("\\[", " [ ");
  expression = expression.replaceAll("\\]", " ] ");
  expression = expression.replaceAll("\\(", " ( ");
  expression = expression.replaceAll("\\)", " ) ");
  expression = expression.replaceAll("\\{", " { ");
  expression = expression.replaceAll("#", " # ");
  expression = expression.replaceAll("}", " } ");
  expression = expression.replaceAll(",", " , ");
  expression = expression.replaceAll("\"", " \" ");
  expression = expression.replaceAll("\\+", " + ");
  expression = expression.replaceAll("-", " - ");
  expression = expression.replaceAll("\\?", " ? ");
  expression = expression.replaceAll("<", " < ");
  expression = expression.replaceAll("<  <", " << ");
  expression = expression.replaceAll(">", " > ");
  expression = expression.replaceAll(">  >", " >> ");
  expression = expression.replaceAll("<  =", " <= ");
  expression = expression.replaceAll(">  =", " >= ");
  expression = expression.replaceAll("-  >", " -> ");

  return expression;
}

/*
 * Verifis whether name is valid
 */
static public boolean validName(String name) {
  if (name.equals(""))
    return false;
  for (int i = 0; i < name.length(); i++) {
    char c = name.charAt(i);
    if (!(c >= 'a' && c <= 'z') && !(c >= 'A' && c <= 'Z')
        && !(c >= '0' && c <= '9' && i > 0) && !(c == '_'))
      return false;
  }
  return true;
}

/*
 * Verifies whether type is valid
 */
static public boolean validType(String name) {
  name = name.replaceAll(" ", "");
  if (name.length() < 5)
    return false;

  if (!name.substring(0, 2).equals("ns"))
    return false;
  if (!(name.charAt(2) == '(')
      || !(name.charAt(name.length() - 1) == ')'))
    return false;

  try {
    Integer.parseInt(name.substring(3, name.length() - 1));
  } catch (Exception e) {
    return false;
```

```
    }

    return true;
  }
}
```

B.6 Package maaoha.view

B.6.1 AddDatapathElementDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.tools.MaaohaTools;
import maaoha.view.model.ModuleView;

/*
 * Class implements dialog for addition of datapath element
 */
public class AddDatapathElementDialog extends JDialog {
  static final long serialVersionUID = 1;

  private String nameText = null;
  private String typeText = null;
  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public AddDatapathElementDialog(Frame aFrame, final String type,
      final ModuleView module, final Point point) {
    super(aFrame, true);
    if (!(point == null))
      setLocation(point.x, point.y);

    setTitle("New " + type + "...");
    final String msgString1 = "name:";
    final JTextField nameField = new JTextField(10);
    final String msgString2 = "type:";
    final JTextField typeField = new JTextField(10);

    Object[] array = { msgString1, nameField, msgString2, typeField };

    final String btnString1 = "Create";
    final String btnString2 = "Cancel";

    Object[] options = { btnString1, btnString2 };

    optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
        JOptionPane.YES_NO_OPTION, null, options, options[0]);
    setContentPane(optionPane);
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
```

```
  public void windowClosing(WindowEvent we) {
    optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
  }
});

nameField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

typeField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

optionPane.addPropertyChangeListener(new PropertyChangeListener() {
  public void propertyChange(PropertyChangeEvent e) {
    String prop = e.getPropertyName();

    if (isVisible()
        && (e.getSource() == optionPane)
        && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
            .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
      Object value = optionPane.getValue();

      if (value == JOptionPane.UNINITIALIZED_VALUE) {
        return;
      }
      optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

      if (value.equals(btnString1)) {
        nameText = nameField.getText();
        typeText = typeField.getText();

        if (!MaaohaTools.validName(nameText)) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(
              AddDatapathElementDialog.this, "Sorry, \""
                  + nameText + "\" "
                  + "is not a valid name.\n"
                  + "Please enter a different name.",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else if (module.getModule().getDatapath()
            .elementExists(nameText)) {
          // name already in use
          nameField.selectAll();
          JOptionPane.showMessageDialog(
              AddDatapathElementDialog.this, "Sorry, \""
                  + nameText + "\" "
                  + "already in use.\n"
                  + "Please enter a different name.",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else if (!MaaohaTools.validType(typeText)) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(
              AddDatapathElementDialog.this, "Sorry, \""
                  + typeText + "\" "
                  + "is not a valid type.\n",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else {
          // we're done; dismiss the dialog
```

```
              try {
                module.addNewElement(point, nameText, type,
                    typeText);
              } catch (MaaohaException ex) {
                System.out.println(ex);
                System.exit(1);
              }
              setVisible(false);
            }
          } else {
            setVisible(false);
          }
        }
      }
    });
  }
}
```

B.6.2 AddExpressionDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

import maaoha.MaaohaException;
import maaoha.view.model.ModuleView;
import maaoha.view.model.SfgView;

/*
 * Class implements dialog for addition of expression
 */
public class AddExpressionDialog extends JDialog {
  static final long serialVersionUID = 1;

  private SfgView selectedSfg = null;
  private String expressionText = null;
  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public AddExpressionDialog(Frame aFrame, final ModuleView module,
      final Point point) {
    super(aFrame, true);

    setLocation(point.x, point.y);
    setTitle("New expression...");

    final String msgString1 = "signal flow graph:";
    final JComboBox sfgCombo = new JComboBox();
    final String msgString2 = "expressions:";
    final JTextArea expressionField = new JTextArea(5, 30);
    final JScrollPane scrollPane = new JScrollPane(expressionField);
```

```
  int index = 0;

  for (int i = 0; i < module.getModule().getDatapath().getSfgsNo(); i++) {
    SfgView sfg = module.getDatapath().getSfg(i);
    sfgCombo.addItem(sfg);
    if (sfg == module.getDatapath().getCurrentView())
      index = i;
  }

  sfgCombo.setSelectedIndex(index);

  Object[] array = { msgString1, sfgCombo, msgString2, scrollPane };

  final String btnString1 = "Create";
  final String btnString2 = "Cancel";

  Object[] options = { btnString1, btnString2 };

  optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
      JOptionPane.YES_NO_OPTION, null, options, options[0]);
  setContentPane(optionPane);
  setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
      optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
    }
  });

  sfgCombo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
  });

  optionPane.addPropertyChangeListener(new PropertyChangeListener() {
    public void propertyChange(PropertyChangeEvent e) {
      String prop = e.getPropertyName();

      if (isVisible()
          && (e.getSource() == optionPane)
          && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
              .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
        Object value = optionPane.getValue();

        if (value == JOptionPane.UNINITIALIZED_VALUE) {
          return;
        }

        optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

        if (value.equals(btnString1)) {
          selectedSfg = (SfgView) sfgCombo.getSelectedItem();
          expressionText = expressionField.getText();
          try {
            module.addNewExpression(point, expressionText,
                selectedSfg);
          } catch (MaaohaException ex) {
            JOptionPane.showMessageDialog(
                AddExpressionDialog.this,
                "Invalid expression:\n" + expressionText,
                "Try again", JOptionPane.ERROR_MESSAGE);
            expressionText = null;
          }
        }
        setVisible(false);
      }
    }
  });
}
```

```
}
```

B.6.3 AddFsmStateDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.tools.MaaohaTools;
import maaoha.view.model.ModuleView;

/*
 * Class implements dialog for addition of controller state
 */
public class AddFsmStateDialog extends JDialog {
  static final long serialVersionUID = 1;

  private String nameText = null;
  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public AddFsmStateDialog(Frame aFrame, final ModuleView module,
      final Point point) {
    super(aFrame, true);

    if (!(point == null))
      setLocation(point.x, point.y);

    setTitle("New state...");

    final String msgString1 = "name:";
    final JTextField nameField = new JTextField(10);

    Object[] array = { msgString1, nameField };

    final String btnString1 = "Create";
    final String btnString2 = "Cancel";

    Object[] options = { btnString1, btnString2 };

    optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
        JOptionPane.YES_NO_OPTION, null, options, options[0]);
    setContentPane(optionPane);
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
      }
    });

    nameField.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
```

```
        optionPane.setValue(btnString1);
      }
    });

    optionPane.addPropertyChangeListener(new PropertyChangeListener() {
      public void propertyChange(PropertyChangeEvent e) {
        String prop = e.getPropertyName();

        if (isVisible()
            && (e.getSource() == optionPane)
            && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
                .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
          Object value = optionPane.getValue();

          if (value == JOptionPane.UNINITIALIZED_VALUE) {
            return;
          }
          optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

          if (value.equals(btnString1)) {
            nameText = nameField.getText();

            if (module.getModule().getFsm().stateExist(nameText)) {
              // text was invalid
              nameField.selectAll();
              JOptionPane.showMessageDialog(
                  AddFsmStateDialog.this, "Sorry, \""
                      + nameText + "\" "
                      + "already in use.\n"
                      + "Please enter a different name.",
                  "Try again", JOptionPane.ERROR_MESSAGE);
              nameText = null;
            } else if (!MaaohaTools.validName(nameText)) {
              // text was invalid
              nameField.selectAll();
              JOptionPane.showMessageDialog(
                  AddFsmStateDialog.this, "Sorry, \""
                      + nameText + "\" "
                      + "is not a valid name.\n"
                      + "Please enter a different name.",
                  "Try again", JOptionPane.ERROR_MESSAGE);
              nameText = null;
            } else {
              // we're done; dismiss the dialog
              try {
                module.addNewFsmState(point, nameText);
              } catch (MaaohaException ex) {
                System.out.println(ex);
                System.exit(1);
              }
              setVisible(false);
            }
          } else {
            setVisible(false);
          }
        }
      }
    });
  }

}
```

B.6.4 AddFsmTransitionDialog.java

```
package maaoha.view;

import java.awt.Frame;
```

```
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.view.model.FsmStateView;
import maaoha.view.model.ModuleView;

/*
 * Class implements dialog for addition of controller transition
 */
public class AddFsmTransitionDialog extends JDialog {
  static final long serialVersionUID = 1;

  private FsmStateView selectedSource = null;
  private FsmStateView selectedTarget = null;
  private String conditionsText = null;
  private String sfgsText = null;
  private JOptionPane optionPane;
  private JTextField conditionsField = null;
  private JTextField sfgsField = null;

  /*
   * Constructor
   */
  public AddFsmTransitionDialog(Frame aFrame, final ModuleView moduleView,
      final Point point) {
    super(aFrame, true);

    if (point != null)
      setLocation(point.x, point.y);
    setTitle("New transition...");

    final String msgString1 = "source:";
    final JComboBox moduleCombo1 = new JComboBox();
    final String msgString2 = "destination:";
    final JComboBox moduleCombo2 = new JComboBox();
    final String msgString3 = "condition:";
    final String msgString4 = "sfgs:";

    conditionsField = new JTextField(30);
    sfgsField = new JTextField(30);

    int index = 0;

    for (int i = 0; i < moduleView.getFsmV().getFsmStateNo(); i++) {
      FsmStateView stateV = moduleView.getFsmV().getFsmState(i);
      moduleCombo1.addItem(stateV.getFsmState().getName());
      moduleCombo2.addItem(stateV.getFsmState().getName());
    }

    moduleCombo1.setSelectedIndex(index);

    Object[] array = { msgString1, moduleCombo1, msgString2, moduleCombo2,
        msgString3, conditionsField, msgString4, sfgsField };

    final String btnString1 = "Create";
    final String btnString2 = "Cancel";
```

```
Object[] options = { btnString1, btnString2 };

optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
    JOptionPane.YES_NO_OPTION, null, options, options[0]);
setContentPane(optionPane);
setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent we) {
    optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
  }
});

conditionsField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

sfgsField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

optionPane.addPropertyChangeListener(new PropertyChangeListener() {
  public void propertyChange(PropertyChangeEvent e) {
    String prop = e.getPropertyName();

    if (isVisible()
        && (e.getSource() == optionPane)
        && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
            .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
      Object value = optionPane.getValue();

      if (value == JOptionPane.UNINITIALIZED_VALUE) {
        return;
      }

      optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

      if (value.equals(btnString1)) {
        selectedSource = moduleView.getFsmV().getFsmState(
            moduleCombo1.getSelectedIndex());
        selectedTarget = moduleView.getFsmV().getFsmState(
            moduleCombo2.getSelectedIndex());
        conditionsText = conditionsField.getText();
        sfgsText = sfgsField.getText();

        if (conditionsText.equals(""))
          conditionsText = "1";

        // we're done; dismiss the dialog
        try {
          moduleView.addNewTransition(selectedSource,
              selectedTarget, conditionsText, sfgsText);
          if (moduleView.getParent().getToolView()
              .getVerifyModel()) {
            try {
              moduleView.getParent().checkModel();
            } catch (MaaohaException ex) {
              ex.toString();
            }
          }
        } catch (MaaohaException ex) {
          System.out.println(ex);
          System.exit(1);
        }
        setVisible(false);
```

```
          } else {
            setVisible(false);
          }
        }
      }
    });
  }
}
```

B.6.5 AddModuleDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.tools.MaaohaTools;
import maaoha.view.model.ModelView;

/*
 * Class implements dialog for addition of module
 */
public class AddModuleDialog extends JDialog {
  static final long serialVersionUID = 1;

  private String nameText = null;
  private String expressionsText = null;
  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public AddModuleDialog(Frame aFrame, final ModelView model, Point point) {
    super(aFrame, true);

    if (point != null)
      setLocation(point.x, point.y);
    setTitle("New module...");

    final String msgString1 = "name:";
    final JTextField nameField = new JTextField(10);

    Object[] array = { msgString1, nameField };

    final String btnString1 = "Create";
    final String btnString2 = "Cancel";

    Object[] options = { btnString1, btnString2 };

    optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
        JOptionPane.YES_NO_OPTION, null, options, options[0]);
    setContentPane(optionPane);
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
```

```
  }
});

nameField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

optionPane.addPropertyChangeListener(new PropertyChangeListener() {
  public void propertyChange(PropertyChangeEvent e) {
    String prop = e.getPropertyName();

    if (isVisible()
        && (e.getSource() == optionPane)
        && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
            .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
      Object value = optionPane.getValue();

      if (value == JOptionPane.UNINITIALIZED_VALUE) {
        // ignore reset
        return;
      }

      optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

      if (value.equals(btnString1)) {
        nameText = nameField.getText();

        if (!MaaohaTools.validName(nameText)) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(AddModuleDialog.this,
              "Sorry, \"" + nameText + "\" "
                  + "is not a valid name.\n"
                  + "Please enter a different name.",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else if (model.getModel().getModule(nameText) != null) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(AddModuleDialog.this,
              "Sorry, \"" + nameText + "\" "
                  + "already in use.\n"
                  + "Please enter a different name.",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else {
          // we're done; dismiss the dialog
          try {
            model.createModule(nameText);
            model.getToolView().getToolBar()
                .actionsEnableModulePresent();
          } catch (MaaohaException ex) {
            // text was invalid
            nameField.selectAll();
            JOptionPane.showMessageDialog(
                AddModuleDialog.this,
                "Sorry, not a valid expression(s):\n"
                    + expressionsText, "Try again",
                JOptionPane.ERROR_MESSAGE);
            nameText = null;
          }
          setVisible(false);
        }
      } else {
        setVisible(false);
      }
```

```
        }
      }
    });
  }
}
```

B.6.6 AddSfgDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.tools.MaaohaTools;
import maaoha.view.model.ModuleView;

/*
 * Class implements dialog for addition of signal flow graph
 */
public class AddSfgDialog extends JDialog {
  static final long serialVersionUID = 1;

  private String nameText = null;
  private String expressionsText = null;
  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public AddSfgDialog(Frame aFrame, final ModuleView module, Point point) {
    super(aFrame, true);
    if (point != null)
      setLocation(point.x, point.y);
    setTitle("New signal flow graph...");

    final String msgString1 = "name:";
    final JTextField nameField = new JTextField(10);
    final String msgString2 = "expressions:";

    final JTextArea expressionsArea = new JTextArea(5, 10);
    final JScrollPane scrollPane = new JScrollPane(expressionsArea);

    Object[] array = { msgString1, nameField, msgString2, scrollPane };

    final String btnString1 = "Create";
    final String btnString2 = "Cancel";

    Object[] options = { btnString1, btnString2 };

    optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
        JOptionPane.YES_NO_OPTION, null, options, options[0]);
    setContentPane(optionPane);
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
```

```
  public void windowClosing(WindowEvent we) {
    optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
  }
});

nameField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

optionPane.addPropertyChangeListener(new PropertyChangeListener() {
  public void propertyChange(PropertyChangeEvent e) {
    String prop = e.getPropertyName();

    if (isVisible()
        && (e.getSource() == optionPane)
        && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
            .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
      Object value = optionPane.getValue();

      if (value == JOptionPane.UNINITIALIZED_VALUE) {
        // ignore reset
        return;
      }

      optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

      if (value.equals(btnString1)) {
        nameText = nameField.getText();
        expressionsText = expressionsArea.getText();

        if (!MaaohaTools.validName(nameText)) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(AddSfgDialog.this,
              "Sorry, \"" + nameText + "\" "
                  + "is not a valid name.\n"
                  + "Please enter a different name.",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else if (module.getModule().getDatapath()
            .elementExists(nameText)) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(AddSfgDialog.this,
              "Sorry, \"" + nameText + "\" "
                  + "already in use.\n"
                  + "Please enter a different name.",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else {
          // we're done; dismiss the dialog
          try {
            module.getDatapath().createSfg(nameText,
                expressionsText);
            module.getParent().getToolView().getToolBar()
                .actionsEnableSfgPresent();
          } catch (MaaohaException ex) {
            // text was invalid
            nameField.selectAll();
            JOptionPane.showMessageDialog(
                AddSfgDialog.this,
                "Sorry, not a valid expression(s):\n"
                    + expressionsText, "Try again",
                JOptionPane.ERROR_MESSAGE);
            nameText = null;
          }
```

```
              setVisible(false);
            }
          } else {
            setVisible(false);
          }
        }
      }
    });
  }
}
```

B.6.7 ControllerGraphSelectionListener.java

```
package maaoha.view;

import maaoha.view.model.FsmTransitionView;
import maaoha.view.model.ModuleView;

import org.jgraph.event.GraphSelectionEvent;
import org.jgraph.event.GraphSelectionListener;
import org.jgraph.graph.DefaultEdge;

/*
 * Class implements controller graph selection listener
 */
public class ControllerGraphSelectionListener implements GraphSelectionListener {
  protected ModuleView moduleView;

  /*
   * Constructor
   */
  public ControllerGraphSelectionListener(ModuleView moduleView) {
    this.moduleView = moduleView;
  }

  /*
   * (non-Javadoc)
   *
   * @see
org.jgraph.event.GraphSelectionListener#valueChanged(org.jgraph.event.GraphSelectio
nEvent)
   */
  public void valueChanged(GraphSelectionEvent e) {
    Object cell = e.getCell();
    if (cell instanceof DefaultEdge) {
      Object obj = ((DefaultEdge) cell).getUserObject();
      if (obj instanceof FsmTransitionView) {
        if (moduleView.getParent().getToolView().getListenersOn()) {
          moduleView.getParent().getToolView().setListenersOn(false);
          moduleView.getDatapath().setCurrentViewTransition(
              (FsmTransitionView) obj);
          moduleView.getParent().getToolView().getToolBar()
              .setViewTypeCombo(obj);
        }
        moduleView.getParent().getToolView().setListenersOn(true);
      }
    }
  }
}
```

B.6.8 DatapathGraph.java

```
package maaoha.view;

import org.jgraph.JGraph;
```

```
import org.jgraph.graph.GraphLayoutCache;
import org.jgraph.graph.GraphModel;

/*
 * Class implements datapath graph
 */
public class DatapathGraph extends JGraph {
  static final long serialVersionUID = 1;

  /*
   * Constructs datapath graph using the model
   */
  public DatapathGraph(GraphModel model) {
    this(model, null);
  }

  /*
   * Constructs datapath graph using the model
   */
  public DatapathGraph(GraphModel model, GraphLayoutCache cache) {
    super(model, cache);
  }
}
```

B.6.9 EditDatapathElementDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import maaoha.tools.MaaohaTools;
import maaoha.view.model.DatapathElementView;
import maaoha.view.model.ModuleView;

/*
 * Class implements dialog for edition of datapath element
 */
public class EditDatapathElementDialog extends JDialog {

  static final long serialVersionUID = 1;

  private String nameText = null;
  private String typeText = null;
  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public EditDatapathElementDialog(Frame aFrame, final String type,
      final ModuleView module, final Point point,
      final DatapathElementView dev) {
    super(aFrame, true);

    if (!(point == null))
      setLocation(point.x, point.y);
```

```
setTitle("Edit " + type + "...");

final String msgString1 = "name:";
final JTextField nameField = new JTextField(10);
nameField.setText(dev.getElement().getName());
final String msgString2 = "type:";
final JTextField typeField = new JTextField(10);
typeField.setText(dev.getElement().getValueType());

Object[] array = { msgString1, nameField, msgString2, typeField };

final String btnString1 = "Ok";
final String btnString2 = "Cancel";

Object[] options = { btnString1, btnString2 };

optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
    JOptionPane.YES_NO_OPTION, null, options, options[0]);
setContentPane(optionPane);
setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent we) {
    optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
  }
});

nameField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

typeField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

optionPane.addPropertyChangeListener(new PropertyChangeListener() {
  public void propertyChange(PropertyChangeEvent e) {
    String prop = e.getPropertyName();

    if (isVisible()
        && (e.getSource() == optionPane)
        && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
            .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
      Object value = optionPane.getValue();

      if (value == JOptionPane.UNINITIALIZED_VALUE) {
        // ignore reset
        return;
      }

      optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

      if (value.equals(btnString1)) {
        nameText = nameField.getText();
        typeText = typeField.getText();

        if (!MaaohaTools.validName(nameText)) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(
              EditDatapathElementDialog.this, "Sorry, \""
                  + nameText + "\" "
                  + "is not a valid name.\n"
                  + "Please enter a different name.",
```

```
                    "Try again", JOptionPane.ERROR_MESSAGE);
                nameText = null;
              } else if (module.getModule().getDatapath()
                  .elementExists(nameText)) {
                // name already in use
                nameField.selectAll();
                JOptionPane.showMessageDialog(
                    EditDatapathElementDialog.this, "Sorry, \""
                        + nameText + "\" "
                        + "already in use.\n"
                        + "Please enter a different name.",
                    "Try again", JOptionPane.ERROR_MESSAGE);
                nameText = null;
              } else if (!MaaohaTools.validType(typeText)) {
                // text was invalid
                nameField.selectAll();
                JOptionPane.showMessageDialog(
                    EditDatapathElementDialog.this, "Sorry, \""
                        + typeText + "\" "
                        + "is not a valid type.\n",
                    "Try again", JOptionPane.ERROR_MESSAGE);
                nameText = null;
              } else {
                // we're done; dismiss the dialog
                dev.getElement().setName(nameText);
                dev.getElement().setValueType(typeText);
                setVisible(false);
              }
            } else {
              setVisible(false);
            }
          }
        }
      });
    }
}
```

B.6.10 EditExpressionDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.view.model.ExpressionView;
import maaoha.view.model.ModuleView;
import maaoha.view.model.SfgView;

/*
 * Class implements dialog for edition of expressions
 */
public class EditExpressionDialog extends JDialog {
  static final long serialVersionUID = 1;
```

```
private SfgView selectedSfg = null;
private String expressionText = null;
private JOptionPane optionPane;

/*
 * Constructor
 */
public EditExpressionDialog(Frame aFrame, final ModuleView module,
    final Point point, final ExpressionView expressionView) {
  super(aFrame, true);
  setLocation(point.x, point.y);
  setTitle("Edit expression...");

  final String msgString1 = "signal flow graph:";
  final JComboBox sfgCombo = new JComboBox();
  final String msgString2 = "expression:";
  final JTextField expressionField = new JTextField(30);
  final JScrollPane scrollPane = new JScrollPane(expressionField);
  expressionField
      .setText(expressionView.getExpression().getValue() + ";");
  int index = 0;

  for (int i = 0; i < module.getModule().getDatapath().getSfgsNo(); i++) {
    SfgView sfg = module.getDatapath().getSfg(i);
    sfgCombo.addItem(sfg);
    if (sfg == expressionView.getParent())
      index = i;
  }

  sfgCombo.setSelectedIndex(index);

  Object[] array = { msgString1, sfgCombo, msgString2, scrollPane };

  final String btnString1 = "Ok";
  final String btnString2 = "Cancel";

  Object[] options = { btnString1, btnString2 };

  optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
      JOptionPane.YES_NO_OPTION, null, options, options[0]);
  setContentPane(optionPane);
  setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
      optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
    }
  });

  sfgCombo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
  });

  optionPane.addPropertyChangeListener(new PropertyChangeListener() {
    public void propertyChange(PropertyChangeEvent e) {
      String prop = e.getPropertyName();

      if (isVisible()
          && (e.getSource() == optionPane)
          && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
              .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
        Object value = optionPane.getValue();

        if (value == JOptionPane.UNINITIALIZED_VALUE) {
          // ignore reset
          return;
        }
```

```
          optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

          if (value.equals(btnString1)) {
            selectedSfg = (SfgView) sfgCombo.getSelectedItem();
            expressionText = expressionField.getText();

            // we're done; dismiss the dialog
            try {
              module.addNewExpression(point, expressionText,
                  selectedSfg);
            } catch (MaaohaException ex) {
              JOptionPane.showMessageDialog(
                  EditExpressionDialog.this,
                  "Invalid expression:\n" + expressionText,
                  "Try again", JOptionPane.ERROR_MESSAGE);
              expressionText = null;
            }
            setVisible(false);
          } else {
            setVisible(false);
          }
        }
      }
    });
  }
}
```

B.6.11 EditFsmTransistionDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.view.model.FsmStateView;
import maaoha.view.model.FsmTransitionView;
import maaoha.view.model.ModuleView;

/*
 * Class implements dialog for edition of controller transition
 */
public class EditFsmTransitionDialog extends JDialog {
  static final long serialVersionUID = 1;

  private FsmStateView selectedSource = null;
  private FsmStateView selectedTarget = null;
  private String conditionsText = null;
  private String sfgsText = null;
  private JOptionPane optionPane;
  JTextField conditionsField = null;
  JTextField sfgsField = null;

  /*
   * Constructor
   */
```

```
public EditFsmTransitionDialog(Frame aFrame,
    final FsmTransitionView transView, final Point point) {
  super(aFrame, true);
  final ModuleView moduleView = transView.getParent().getParent()
      .getParent();

  if (point != null)
    setLocation(point.x, point.y);
  setTitle("Edit transition...");

  final String msgString1 = "source:";
  final JComboBox moduleCombo1 = new JComboBox();
  final String msgString2 = "destination:";
  final JComboBox moduleCombo2 = new JComboBox();
  final String msgString3 = "condition:";
  final String msgString4 = "sfgs:";

  conditionsField = new JTextField(30);
  sfgsField = new JTextField(30);

  int sourceIndex = 0;
  int targetIndex = 0;

  for (int i = 0; i < moduleView.getFsmV().getFsmStateNo(); i++) {
    FsmStateView stateV = moduleView.getFsmV().getFsmState(i);
    moduleCombo1.addItem(stateV.getFsmState().getName());
    moduleCombo2.addItem(stateV.getFsmState().getName());
    if (stateV == transView.getParent())
      sourceIndex = i;
    if (stateV == moduleView.getFsmV().getFsmStateView(
        transView.getTransition().getNextState()))
      targetIndex = i;
  }

  moduleCombo1.setSelectedIndex(sourceIndex);
  moduleCombo2.setSelectedIndex(targetIndex);
  conditionsField.setText(transView.getTransition().getCondition()
      .getBasicExpression().toString());
  sfgsField.setText(transView.getTransition().getSfgsString());

  Object[] array = { msgString1, moduleCombo1, msgString2, moduleCombo2,
      msgString3, conditionsField, msgString4, sfgsField };

  final String btnString1 = "Edit";
  final String btnString2 = "Cancel";

  Object[] options = { btnString1, btnString2 };

  optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
      JOptionPane.YES_NO_OPTION, null, options, options[0]);
  setContentPane(optionPane);
  setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
      optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
    }
  });

  conditionsField.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      optionPane.setValue(btnString1);
    }
  });

  sfgsField.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      optionPane.setValue(btnString1);
    }
```

```
    });

    optionPane.addPropertyChangeListener(new PropertyChangeListener() {
      public void propertyChange(PropertyChangeEvent e) {
        String prop = e.getPropertyName();
        if (isVisible()
            && (e.getSource() == optionPane)
            && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
                .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
          Object value = optionPane.getValue();

          if (value == JOptionPane.UNINITIALIZED_VALUE) {
            // ignore reset
            return;
          }
          optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

          if (value.equals(btnString1)) {
            selectedSource = moduleView.getFsmV().getFsmState(
                moduleCombo1.getSelectedIndex());
            selectedTarget = moduleView.getFsmV().getFsmState(
                moduleCombo2.getSelectedIndex());
            conditionsText = conditionsField.getText();
            sfgsText = sfgsField.getText();

            if (conditionsText.equals(""))
              conditionsText = "1";
            // we're done; dismiss the dialog
            try {
              transView.getParent().getParent().editTransition(
                  selectedSource, selectedTarget,
                  conditionsText, sfgsText, transView);
              if (moduleView.getParent().getToolView()
                  .getVerifyModel()) {
                try {
                  moduleView.getParent().checkModel();
                } catch (MaaohaException ex) {
                  ex.toString();
                }
              }
            } catch (MaaohaException ex) {
              System.out.println(ex);
              System.exit(1);
            }
            setVisible(false);
          } else {
            setVisible(false);
          }
        }
      }
    });
  }
}
```

B.6.12 EditSfgDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
```

```
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.model.Expression;
import maaoha.model.Sfg;
import maaoha.tools.MaaohaTools;
import maaoha.view.model.ModuleView;
import maaoha.view.model.SfgView;

/*
 * Class implements dialog for edition of signal flow graph
 */
public class EditSfgDialog extends JDialog {
  static final long serialVersionUID = 1;

  private String nameText = null;
  private String expressionsText = null;
  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public EditSfgDialog(Frame aFrame, final ModuleView module, Point point) {
    super(aFrame, true);
    if (point != null)
      setLocation(point.x, point.y);
    setTitle("Edit signal flow graph...");

    final String msgString0 = "old name:";
    final JComboBox sfgCombo = new JComboBox();
    int index = 0;

    for (int i = 0; i < module.getModule().getDatapath().getSfgsNo(); i++) {
      SfgView sfg = module.getDatapath().getSfg(i);
      sfgCombo.addItem(sfg);
      if (sfg == module.getDatapath().getCurrentView())
        index = i;
    }
    Sfg sfg = module.getDatapath().getSfg(index).getSfg();
    final String msgString1 = "name:";
    final JTextField nameField = new JTextField(10);
    nameField.setText(sfg.getName());
    final String msgString2 = "expressions:";

    final JTextArea expressionsArea = new JTextArea(5, 10);
    String str = "";
    for (int i = 0; i < sfg.getExpressionsNo(); i++) {
      Expression ev = sfg.getExpression(i);
      str += ev.getValue() + ";\n";
    }
    expressionsArea.setText(str);
    final JScrollPane scrollPane = new JScrollPane(expressionsArea);

    Object[] array = { msgString0, sfgCombo, msgString1, nameField,
        msgString2, scrollPane };

    final String btnString1 = "Ok";
    final String btnString2 = "Cancel";

    Object[] options = { btnString1, btnString2 };

    optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
        JOptionPane.YES_NO_OPTION, null, options, options[0]);
```

```
setContentPane(optionPane);
setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent we) {
    optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
  }
});

nameField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

sfgCombo.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    Sfg sfg = module.getDatapath().getSfg(
        sfgCombo.getSelectedIndex()).getSfg();
    nameField.setText(sfg.getName());
    String str = "";
    for (int i = 0; i < sfg.getExpressionsNo(); i++) {
      Expression ev = sfg.getExpression(i);
      str += ev.getValue() + ";\n";
    }
    expressionsArea.setText(str);
  }
});

optionPane.addPropertyChangeListener(new PropertyChangeListener() {
  public void propertyChange(PropertyChangeEvent e) {
    String prop = e.getPropertyName();

    if (isVisible()
        && (e.getSource() == optionPane)
        && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
            .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
      Object value = optionPane.getValue();

      if (value == JOptionPane.UNINITIALIZED_VALUE) {
        // ignore reset
        return;
      }

      optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

      if (value.equals(btnString1)) {
        nameText = nameField.getText();
        expressionsText = expressionsArea.getText();

        if (!MaaohaTools.validName(nameText)) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(EditSfgDialog.this,
              "Sorry, \"" + nameText + "\" "
                  + "is not a valid name.\n"
                  + "Please enter a different name.",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else if (module.getModule().getDatapath()
            .elementExists(nameText)
            && !nameText.equals(module.getDatapath()
                .getSfg(sfgCombo.getSelectedIndex())
                .getSfg().getName())) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(EditSfgDialog.this,
              "Sorry, \"" + nameText + "\" "
                  + "already in use.\n"
```

```
                    + "Please enter a different name.",
                "Try again", JOptionPane.ERROR_MESSAGE);
            nameText = null;
          } else {
            // we're done; dismiss the dialog
            try {
              module.getDatapath().editSfg(
                  nameText,
                  expressionsText,
                  module.getDatapath().getSfg(
                      sfgCombo.getSelectedIndex()));
            } catch (MaaohaException ex) {
              // text was invalid
              JOptionPane.showMessageDialog(
                  EditSfgDialog.this,
                  "Sorry, not a valid expression(s):\n"
                      + expressionsText, "Try again",
                  JOptionPane.ERROR_MESSAGE);
              nameText = null;
            }
            setVisible(false);
          }
        } else {
          setVisible(false);
        }
      }
    }
  });
  }
}
```

B.6.13 EditUseModuleDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.model.Module;
import maaoha.tools.MaaohaTools;
import maaoha.view.model.ModuleView;
import maaoha.view.model.UseModuleView;

/*
 * Class implements dialog for edition of sub-module
 */
public class EditUseModuleDialog extends JDialog {
  static final long serialVersionUID = 1;

  private Module selectedModule = null;
  private String parametersText = null;
  private String nameText = null;
  private JOptionPane optionPane;
  private JTextField parametersField = null;
  private JTextField nameField = null;
```

```
/*
 * Constructor
 */
public EditUseModuleDialog(Frame aFrame, final ModuleView moduleView,
    final Point point, final UseModuleView umv) {
  super(aFrame, true);
  if (point != null)
    setLocation(point.x, point.y);
  setTitle("Edit module...");

  final String msgString0 = "Module name:";
  nameField = new JTextField(30);
  nameField.setText(umv.getUseModule().getName());
  final String msgString1 = "Use module:";
  final JComboBox moduleCombo = new JComboBox();
  final String msgString2 = "Parameters:";
  parametersField = new JTextField(30);

  int index = 0;

  for (int i = 0; i < moduleView.getParent().getModel().getModulesNo(); i++) {
    Module module = moduleView.getParent().getModel().getModule(i);
    moduleCombo.addItem(module.getName());
    if (module == umv.getUseModule().getModule())
      index = i;
  }

  moduleCombo.setSelectedIndex(index);

  parametersField.setText(umv.getUseModule().parametersString());

  Object[] array = { msgString0, nameField, msgString1, moduleCombo,
      msgString2, parametersField };

  final String btnString1 = "Ok";
  final String btnString2 = "Cancel";

  Object[] options = { btnString1, btnString2 };

  optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
      JOptionPane.YES_NO_OPTION, null, options, options[0]);
  setContentPane(optionPane);
  setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
      optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
    }
  });

  moduleCombo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
  });

  parametersField.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      optionPane.setValue(btnString1);
    }
  });

  nameField.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      optionPane.setValue(btnString1);
    }
  });

  optionPane.addPropertyChangeListener(new PropertyChangeListener() {
```

```
        public void propertyChange(PropertyChangeEvent e) {
          String prop = e.getPropertyName();

          if (isVisible()
              && (e.getSource() == optionPane)
              && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
                  .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
            Object value = optionPane.getValue();

            if (value == JOptionPane.UNINITIALIZED_VALUE) {
              // ignore reset
              return;
            }

            optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

            if (value.equals(btnString1)) {
              selectedModule = moduleView.getParent().getModel()
                  .getModule(moduleCombo.getSelectedIndex());
              parametersText = parametersField.getText();
              nameText = nameField.getText();

              if (!MaaohaTools.validName(nameText)) {
                // text was invalid
                nameField.selectAll();
                JOptionPane.showMessageDialog(
                    EditUseModuleDialog.this, "Sorry, \""
                        + nameText + "\" "
                        + "is not a valid name.\n"
                        + "Please enter a different name.",
                    "Try again", JOptionPane.ERROR_MESSAGE);
                nameText = null;
              } else if (!nameText.equals(umv.getUseModule()
                  .getName())
                  && moduleView.getModule().getDatapath()
                      .elementExists(nameText)) {
                // name already in use
                nameField.selectAll();
                JOptionPane.showMessageDialog(
                    EditUseModuleDialog.this, "Sorry, \""
                        + nameText + "\" "
                        + "already in use.\n"
                        + "Please enter a different name.",
                    "Try again", JOptionPane.ERROR_MESSAGE);
                nameText = null;
              } else {
                moduleView.getDatapath().editUseModule(nameText,
                    selectedModule, parametersText, umv);
                if (moduleView.getParent().getToolView()
                    .getVerifyModel()) {
                  try {
                    moduleView.getParent().checkModel();
                  } catch (MaaohaException ex) {
                    ex.toString();
                  }
                }
                setVisible(false);
              }
            } else {
              setVisible(false);
            }
          }
        }
      });
    }
}
```

B.6.14 FsmGraph.java

```
package maaoha.view;

import org.jgraph.JGraph;
import org.jgraph.graph.GraphLayoutCache;
import org.jgraph.graph.GraphModel;

/*
 * Class implements datapath graph
 */
public class FsmGraph extends JGraph {
  static final long serialVersionUID = 1;

  /*
   * Constructs controller graph using the model
   */
  public FsmGraph(GraphModel model) {
    this(model, null);
  }

  /*
   * Constructs controller graph using the model
   */
  public FsmGraph(GraphModel model, GraphLayoutCache cache) {
    super(model, cache);
  }
}
```

B.6.15 GPCellViewFactory.java

```
package maaoha.view;

import java.util.Map;

import org.jgraph.graph.DefaultCellViewFactory;
import org.jgraph.graph.DefaultGraphCell;
import org.jgraph.graph.VertexView;

/*
 * Class implements graph view factory
 */
public class GPCellViewFactory extends DefaultCellViewFactory {
  public static final String VIEW_CLASS_KEY = "viewClassKey";

  static final long serialVersionUID = 1;

  /*
   * Sets view class
   */
  public static final void setViewClass(Map map, String viewClass) {
    map.put(VIEW_CLASS_KEY, viewClass);
  }

  /*
   * (non-Javadoc)
   *
   * @see
org.jgraph.graph.DefaultCellViewFactory#createVertexView(java.lang.Object)
   */
  protected VertexView createVertexView(Object v) {
    try {
      DefaultGraphCell cell = (DefaultGraphCell) v;
      String viewClass = (String) cell.getAttributes()
          .get(VIEW_CLASS_KEY);
      VertexView view = (VertexView) Thread.currentThread()
```

```
          .getContextClassLoader().loadClass(viewClass).newInstance();
      view.setCell(v);
      return view;
    } catch (Exception ex) {
    }
    return super.createVertexView(v);
  }
}
```

B.6.16 JGraphEllipseView.java

```
package maaoha.view;

import java.awt.BasicStroke;
import java.awt.Dimension;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;

import org.jgraph.graph.CellViewRenderer;
import org.jgraph.graph.EdgeView;
import org.jgraph.graph.GraphConstants;
import org.jgraph.graph.VertexRenderer;
import org.jgraph.graph.VertexView;

/**
 * @author Gaudenz Alder
 *
 */
public class JGraphEllipseView extends VertexView {

  /**
   */
  public static class JGraphEllipseRenderer extends VertexRenderer {

    static final long serialVersionUID = 1;

    /**
     * Return a slightly larger preferred size than for a rectangle.
     */
    public Dimension getPreferredSize() {
      Dimension d = super.getPreferredSize();
      d.width += d.width / 8;
      d.height += d.height / 2;
      return d;
    }

    /**
     */
    public void paint(Graphics g) {
      int b = borderWidth;
      Graphics2D g2 = (Graphics2D) g;
      Dimension d = getSize();
      boolean tmp = selected;
      if (super.isOpaque()) {
        g.setColor(super.getBackground());
        if (gradientColor != null && !preview) {
          setOpaque(false);
          g2.setPaint(new GradientPaint(0, 0, getBackground(),
              getWidth(), getHeight(), gradientColor, true));
        }
        g.fillOval(b - 1, b - 1, d.width - b, d.height - b);
      }
      try {
```

```
        setBorder(null);
        setOpaque(false);
        selected = false;
        super.paint(g);
      } finally {
        selected = tmp;
      }
      if (bordercolor != null) {
        g.setColor(bordercolor);
        g2.setStroke(new BasicStroke(b));
        g.drawOval(b - 1, b - 1, d.width - b, d.height - b);
      }
      if (selected) {
        g2.setStroke(GraphConstants.SELECTION_STROKE);
        g.setColor(highlightColor);
        g.drawOval(b - 1, b - 1, d.width - b, d.height - b);
      }
    }
  }

  /**
   */
  public static transient JGraphEllipseRenderer renderer = new
JGraphEllipseRenderer();

  static final long serialVersionUID = 1;

  /**
   */
  public JGraphEllipseView() {
    super();
  }

  /**
   */
  public JGraphEllipseView(Object cell) {
    super(cell);
  }

  /**
   * Returns the intersection of the bounding rectangle and the straight line
   * between the source and the specified point p. The specified point is
   * expected not to intersect the bounds.
   */
  public Point2D getPerimeterPoint(EdgeView edge, Point2D source, Point2D p) {
    Rectangle2D r = getBounds();

    double x = r.getX();
    double y = r.getY();
    double a = (r.getWidth() + 1) / 2;
    double b = (r.getHeight() + 1) / 2;

    // x0,y0 - center of ellipse
    double x0 = x + a;
    double y0 = y + b;

    // x1, y1 - point
    double x1 = p.getX();
    double y1 = p.getY();

    // calculate straight line equation through point and ellipse center
    // y = d * x + h
    double dx = x1 - x0;
    double dy = y1 - y0;

    if (dx == 0)
      return new Point((int) x0, (int) (y0 + b * dy / Math.abs(dy)));
```

```
    double d = dy / dx;
    double h = y0 - d * x0;

    // calculate intersection
    double e = a * a * d * d + b * b;
    double f = -2 * x0 * e;
    double g = a * a * d * d * x0 * x0 + b * b * x0 * x0 - a * a * b * b;

    double det = Math.sqrt(f * f - 4 * e * g);

    // two solutions (perimeter points)
    double xout1 = (-f + det) / (2 * e);
    double xout2 = (-f - det) / (2 * e);
    double yout1 = d * xout1 + h;
    double yout2 = d * xout2 + h;

    double dist1Squared = Math.pow((xout1 - x1), 2)
        + Math.pow((yout1 - y1), 2);
    double dist2Squared = Math.pow((xout2 - x1), 2)
        + Math.pow((yout2 - y1), 2);

    // correct solution
    double xout, yout;

    if (dist1Squared < dist2Squared) {
      xout = xout1;
      yout = yout1;
    } else {
      xout = xout2;
      yout = yout2;
    }

    return getAttributes().createPoint(xout, yout);
  }

  /**
   */
  public CellViewRenderer getRenderer() {
    return renderer;
  }
}
```

B.6.17 MaaohaControllerMarqueeHandler.java

```
package maaoha.view;

import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.MouseEvent;
import java.awt.geom.Point2D;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.AbstractAction;
import javax.swing.JPopupMenu;
import javax.swing.SwingUtilities;

import maaoha.view.model.FsmStateView;
import maaoha.view.model.FsmTransitionView;
import maaoha.view.model.ModuleView;

import org.jgraph.JGraph;
import org.jgraph.graph.BasicMarqueeHandler;
import org.jgraph.graph.DefaultGraphCell;
import org.jgraph.graph.DefaultPort;
import org.jgraph.graph.PortView;
```

```
/*
 *  Class implements marquee handler that connects vertices
 *  and displays popup menus for controller
 */
public class MaaohaControllerMarqueeHandler extends BasicMarqueeHandler {
  protected JGraph graph;

  protected ModuleView module;

  protected Point2D start, current;

  protected PortView port, firstPort;

  /*
   * Constructor
   */
  public MaaohaControllerMarqueeHandler(JGraph graph, ModuleView module) {
    super();
    this.module = module;
    this.graph = graph;
  }

  /*
   * Adds arrows that are conntected to the vertex to the list
   */
  protected void addArrowsToList(ArrayList<DefaultGraphCell> list,
      DefaultGraphCell cell) {
    for (int i = 0; i < cell.getChildCount(); i++) {
      Object obj = cell.getChildAt(i);
      if (obj instanceof DefaultPort) {
        DefaultPort port = (DefaultPort) obj;
        Iterator iter = port.edges();
        while (iter.hasNext()) {
          DefaultGraphCell gcell = (DefaultGraphCell) iter.next();
          if (!list.contains(gcell))
            list.add(gcell);
        }
      }
    }
  }

  /*
   * Creates popup menu
   */
  public JPopupMenu createPopupMenu(final Point pt, final Object cell,
      final ModuleView module) {
    JPopupMenu menu = new JPopupMenu();
    if (cell != null) {
      // Edit
      menu.add(new AbstractAction("Edit") {
        static final long serialVersionUID = 1;

        public void actionPerformed(ActionEvent e) {
          Object obj = ((DefaultGraphCell) cell).getUserObject();
          if (obj instanceof FsmTransitionView) {
            FsmTransitionView transView = (FsmTransitionView) obj;
            EditFsmTransitionDialog cd = new EditFsmTransitionDialog(
                maaoha.MyTool.frame, transView, pt);
            cd.pack();
            cd.setVisible(true);
          } else if (obj instanceof FsmStateView) {

          }
        }
      });
    }
    // Remove
    if (!graph.isSelectionEmpty() || cell != null) {
```

```
      menu.addSeparator();
      menu.add(new AbstractAction("Remove") {
        static final long serialVersionUID = 1;

        public void actionPerformed(ActionEvent e) {
          ArrayList<DefaultGraphCell> cellsList =
            new ArrayList<DefaultGraphCell>();
          Object[] selection = graph.getSelectionCells();
          for (int i = 0; i < graph.getSelectionCount(); i++) {
            Object element = ((DefaultGraphCell) selection[i])
                .getUserObject();
            if (element instanceof FsmTransitionView) {
              FsmTransitionView transView = (FsmTransitionView) element;
              transView.getParent().removeTransition(transView);
              DefaultGraphCell cell = ((FsmTransitionView) element)
                  .getGraphCell();
              cellsList.add(cell);
            } else if (element instanceof FsmStateView) {
              ((FsmStateView) element).remove();
              DefaultGraphCell cell = ((FsmStateView) element)
                  .getGraphCell();
              cellsList.add(cell);
              addArrowsToList(cellsList, cell);
            }
          }

          DefaultGraphCell[] cells = new DefaultGraphCell[cellsList
              .size()];
          for (int i = 0; i < cellsList.size(); i++)
            cells[i] = cellsList.get(i);
          graph.getGraphLayoutCache().remove(cells);
        }
      });
    }
    menu.addSeparator();
    menu.add(new AbstractAction("Add new state") {
      static final long serialVersionUID = 1;

      public void actionPerformed(ActionEvent ev) {
        AddFsmStateDialog cd = new AddFsmStateDialog(
            maaoha.MyTool.frame, module, pt);
        cd.pack();
        cd.setVisible(true);
      }
    });
    menu.add(new AbstractAction("Add new transition") {
      static final long serialVersionUID = 1;

      public void actionPerformed(ActionEvent ev) {
        AddFsmTransitionDialog cd = new AddFsmTransitionDialog(
            maaoha.MyTool.frame, module, pt);
        cd.pack();
        cd.setVisible(true);
      }
    });
    return menu;
  }

  /*
   * (non-Javadoc)
   *
   * @see
org.jgraph.graph.BasicMarqueeHandler#isForceMarqueeEvent(java.awt.event.MouseEvent)
   */
  public boolean isForceMarqueeEvent(MouseEvent e) {
    if (e.isShiftDown())
      return false;
    // If Right Mouse Button we want to Display the PopupMenu
```

```
    if (SwingUtilities.isRightMouseButton(e))
      // Return Immediately
      return true;
    return super.isForceMarqueeEvent(e);
  }

  /*
   * (non-Javadoc)
   *
   * @see
org.jgraph.graph.BasicMarqueeHandler#mousePressed(java.awt.event.MouseEvent)
   */
  public void mousePressed(final MouseEvent e) {
    // If Right Mouse Button
    if (SwingUtilities.isRightMouseButton(e)) {
      // Find Cell in Model Coordinates
      Object cell = graph.getFirstCellForLocation(e.getX(), e.getY());
      module.refreshBothGraphs();
      // Create PopupMenu for the Cell
      JPopupMenu menu = createPopupMenu(e.getPoint(), cell, module);
      // Display PopupMenu
      menu.show(graph, e.getX(), e.getY());
      // Else if in ConnectMode and Remembered Port is Valid
    } else {
      // Call Superclass
      super.mousePressed(e);
    }
  }
}
```

B.6.18 MaaohaDatapathMarqueeHandler.java

```
package maaoha.view;

import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.MouseEvent;
import java.awt.geom.Point2D;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.AbstractAction;
import javax.swing.JPopupMenu;
import javax.swing.SwingUtilities;

import maaoha.MaaohaException;
import maaoha.view.model.DatapathElementView;
import maaoha.view.model.ExpressionView;
import maaoha.view.model.ModuleView;
import maaoha.view.model.UseModuleView;

import org.jgraph.JGraph;
import org.jgraph.graph.BasicMarqueeHandler;
import org.jgraph.graph.DefaultEdge;
import org.jgraph.graph.DefaultGraphCell;
import org.jgraph.graph.DefaultPort;
import org.jgraph.graph.PortView;

/*
 *  Class implements marquee handler that connects vertices
 *  and displays popup menus for datapath
 */
public class MaaohaDatapathMarqueeHandler extends BasicMarqueeHandler {
  protected JGraph graph;

  protected ModuleView moduleView;
```

```
final public int typeEdge = 0;

final public int typeVertex = 1;

protected Point2D start, current;

protected PortView port, firstPort;

/*
 * Constructor
 */
public MaaohaDatapathMarqueeHandler(JGraph graph, ModuleView moduleView) {
  super();
  this.moduleView = moduleView;
  this.graph = graph;
}

/*
 * Adds arrows that are connected to the specified vertex to the list
 */
protected void addArrowsToList(ArrayList<DefaultGraphCell> list,
    DefaultGraphCell cell) {
  for (int i = 0; i < cell.getChildCount(); i++) {
    Object obj = cell.getChildAt(i);
    if (obj instanceof DefaultPort) {
      DefaultPort port = (DefaultPort) obj;
      Iterator iter = port.edges();
      while (iter.hasNext()) {
        DefaultGraphCell gcell = (DefaultGraphCell) iter.next();
        if (!list.contains(gcell))
          list.add(gcell);
      }
    }
  }
}

/*
 * Creates popup menu
 */
public JPopupMenu createPopupMenu(final Point pt, final Object cell) {
  JPopupMenu menu = new JPopupMenu();

  if (cell instanceof DefaultGraphCell && !(cell instanceof DefaultEdge)) {
    menu.add(new AbstractAction("Edit") {
      static final long serialVersionUID = 1;

      public void actionPerformed(ActionEvent e) {
        Object obj = ((DefaultGraphCell) cell).getUserObject();
        if (obj instanceof ExpressionView) {
          ExpressionView expressionView = (ExpressionView) obj;
          EditExpressionDialog cd = new EditExpressionDialog(
              maaoha.MyTool.frame, moduleView, pt,
              expressionView);
          cd.pack();
          cd.setVisible(true);

          ArrayList<DefaultGraphCell> cellsList =
            new ArrayList<DefaultGraphCell>();
          expressionView.getParent().getParent()
              .removeExpression(expressionView);
          DefaultGraphCell cell = expressionView.getGraphCell();
          cellsList.add(cell);
          addArrowsToList(cellsList, cell);
          graph.getGraphLayoutCache().remove(cellsList.toArray());

          if (moduleView.getParent().getToolView()
              .getVerifyModel()) {
            try {
```

```
            moduleView.getParent().checkModel();
          } catch (MaaohaException ex) {
            ex.toString();
          }
        }
        moduleView.getDatapath().refreshView();
      } else if (obj instanceof DatapathElementView) {
        DatapathElementView dev = (DatapathElementView) obj;
        EditDatapathElementDialog cd = new EditDatapathElementDialog(
            maaoha.MyTool.frame, dev.getElement()
                .getElementType(), moduleView, pt, dev);
        cd.pack();
        cd.setVisible(true);
      } else if (obj instanceof UseModuleView) {
        UseModuleView umv = (UseModuleView) obj;
        EditUseModuleDialog cd = new EditUseModuleDialog(
            maaoha.MyTool.frame, moduleView, pt, umv);
        cd.pack();
        cd.setVisible(true);
      }
    }
  });
}
if (doesSelectionContain(cell, typeVertex)) {
  menu.add(new AbstractAction("Remove") {
    static final long serialVersionUID = 1;

    public void actionPerformed(ActionEvent e) {
      ArrayList<DefaultGraphCell> cellsList =
        new ArrayList<DefaultGraphCell>();
      Object[] selection = graph.getSelectionCells();
      for (int i = 0; i < graph.getSelectionCount(); i++) {
        Object element = ((DefaultGraphCell) selection[i])
            .getUserObject();
        if (element instanceof UseModuleView) {
          UseModuleView umv = (UseModuleView) element;
          umv.getParentDatapathView().removeUseModule(umv);
          DefaultGraphCell cell = umv.getGraphCell();
          cellsList.add(cell);
          addArrowsToList(cellsList, cell);
        } else if (element instanceof ExpressionView) {
          ExpressionView expressionView = ((ExpressionView) element);
          expressionView.getParent().getParent()
              .removeExpression(expressionView);
          DefaultGraphCell cell = expressionView
              .getGraphCell();
          cellsList.add(cell);
          addArrowsToList(cellsList, cell);
        } else if (element instanceof DatapathElementView) {
          DatapathElementView dev = (DatapathElementView) element;
          ArrayList<DefaultEdge> edges = dev
              .getParentDatapathView().removeElement(dev);
          if (edges != null)
            cellsList.addAll(edges);

          DefaultGraphCell cell = dev.getGraphCell();
          cellsList.add(cell);
          addArrowsToList(cellsList, cell);
        }
      }
      graph.getGraphLayoutCache().remove(cellsList.toArray());
      if (moduleView.getParent().getToolView().getVerifyModel()) {
        try {
          moduleView.getParent().checkModel();
        } catch (MaaohaException ex) {
          ex.toString();
        }
      }
```

```
      moduleView.getDatapath().refreshView();
    }
  });
}
menu.addSeparator();
menu.add(new AbstractAction("Create new input") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent ev) {
    AddDatapathElementDialog cd = new AddDatapathElementDialog(
        maaoha.MyTool.frame, maaoha.model.Model.typeInput,
        moduleView, pt);
    cd.pack();
    cd.setVisible(true);
  }
});
menu.add(new AbstractAction("Create new output") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent ev) {
    AddDatapathElementDialog cd = new AddDatapathElementDialog(
        maaoha.MyTool.frame, maaoha.model.Model.typeOutput,
        moduleView, pt);
    cd.pack();
    cd.setVisible(true);
  }
});
menu.add(new AbstractAction("Create new signal") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent ev) {
    AddDatapathElementDialog cd = new AddDatapathElementDialog(
        maaoha.MyTool.frame, maaoha.model.Model.typeSignal,
        moduleView, pt);
    cd.pack();
    cd.setVisible(true);
  }
});
menu.add(new AbstractAction("Create new register") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent ev) {
    AddDatapathElementDialog cd = new AddDatapathElementDialog(
        maaoha.MyTool.frame, maaoha.model.Model.typeRegister,
        moduleView, pt);
    cd.pack();
    cd.setVisible(true);
  }
});
menu.addSeparator();
menu.add(new AbstractAction("Use module") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent ev) {
    UseModuleDialog cd = new UseModuleDialog(maaoha.MyTool.frame,
        moduleView, pt);
    cd.pack();
    cd.setVisible(true);
  }
});
menu.addSeparator();
menu.add(new AbstractAction("Create new signal flow graph") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent ev) {
    AddSfgDialog cd = new AddSfgDialog(maaoha.MyTool.frame,
        moduleView, pt);
    cd.pack();
```

```
      cd.setVisible(true);
    }
  });
  menu.add(new AbstractAction("Edit existing signal flow graph") {
    static final long serialVersionUID = 1;

    public void actionPerformed(ActionEvent ev) {
      EditSfgDialog cd = new EditSfgDialog(maaoha.MyTool.frame,
          moduleView, pt);
      cd.pack();
      cd.setVisible(true);
    }
  });
  menu.add(new AbstractAction("Remove signal flow graph") {
    static final long serialVersionUID = 1;

    public void actionPerformed(ActionEvent ev) {
      RemoveSfgDialog cd = new RemoveSfgDialog(maaoha.MyTool.frame,
          moduleView, pt);
      cd.pack();
      cd.setVisible(true);
    }
  });
  menu.addSeparator();
  menu.add(new AbstractAction("Add expression") {
    static final long serialVersionUID = 1;

    public void actionPerformed(ActionEvent ev) {
      AddExpressionDialog cd = new AddExpressionDialog(
          maaoha.MyTool.frame, moduleView, pt);
      cd.pack();
      cd.setVisible(true);
    }
  });
  return menu;
}

/*
 * Verifies whether selection contain model objects
 */
protected boolean doesSelectionContain(Object cell, int type) {
  if (type == typeEdge)
    if (cell instanceof DefaultEdge)
      return true;
  if (type == typeVertex)
    if (cell instanceof DefaultGraphCell
        && !(cell instanceof DefaultEdge))
      return true;

  Object[] selection = graph.getSelectionCells();
  for (int i = 0; i < graph.getSelectionCount(); i++) {
    Object element = selection[i];
    if (type == typeEdge)
      if (element instanceof DefaultEdge)
        return true;
    if (type == typeVertex)
      if (element instanceof DefaultGraphCell
          && !(element instanceof DefaultEdge))
        return true;
  }
  return false;
}

/*
 * (non-Javadoc)
 *
 * @see org.jgraph.graph.BasicMarqueeHandler
 * #isForceMarqueeEvent(java.awt.event.MouseEvent)
```

```
   */
  public boolean isForceMarqueeEvent(MouseEvent e) {
    if (e.isShiftDown())
      return false;
    // If Right Mouse Button we want to Display the PopupMenu
    if (SwingUtilities.isRightMouseButton(e))
      // Return Immediately
      return true;
    // Else Call Superclass
    return super.isForceMarqueeEvent(e);
  }

  /*
   * (non-Javadoc)
   *
   * @see
org.jgraph.graph.BasicMarqueeHandler#mousePressed(java.awt.event.MouseEvent)
   */
  public void mousePressed(final MouseEvent e) {
    // If Right Mouse Button
    if (SwingUtilities.isRightMouseButton(e)) {
      // Find Cell in Model Coordinates
      Object cell = graph.getFirstCellForLocation(e.getX(), e.getY());
      moduleView.refreshBothGraphs();
      // Create PopupMenu for the Cell
      JPopupMenu menu = createPopupMenu(e.getPoint(), cell);
      // Display PopupMenu
      menu.show(graph, e.getX(), e.getY());
    } else {
      // Call Superclass
      super.mousePressed(e);
    }
  }
}
```

B.6.19 MaaohaMenuBar.java

```
package maaoha.view;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

import maaoha.model.Model;

/*
 * Class implements menu bar
 */
public class MaaohaMenuBar extends JMenuBar {
  static final long serialVersionUID = 1;

  /*
   * Constructor
   */
  public MaaohaMenuBar(final ToolView app) {
    JMenu fileMenu = new JMenu("File");

    fileMenu.add(new JMenuItem(new AbstractAction("New model") {
      static final long serialVersionUID = 1;

      public void actionPerformed(ActionEvent e) {
        app.createNewModel();
      }
    }));
```

```
fileMenu.addSeparator();

fileMenu.add(new JMenuItem(new AbstractAction(
    "Import model from XML file") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.readModelFromXML();
  }
}));

fileMenu.add(new JMenuItem(new AbstractAction(
    "Export model to XML file") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.writeModelToXML();
  }
}));

fileMenu.addSeparator();

fileMenu.add(new JMenuItem(new AbstractAction(
    "Import project from XML file") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.readModelViewFromXML();
  }
}));

fileMenu.add(new JMenuItem(new AbstractAction(
    "Export project to XML file") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.writeModelViewToXML();
  }
}));

fileMenu.addSeparator();

fileMenu.add(new JMenuItem(
    new AbstractAction("Import from Gezel file") {
      static final long serialVersionUID = 1;

      public void actionPerformed(ActionEvent e) {
        app.readModelFromGezelSource();
      }
    }));

fileMenu.add(new JMenuItem(new AbstractAction("Export to Gezel file") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.writeModelToGezelSource();
  }
}));

fileMenu.addSeparator();
fileMenu.add(new JMenuItem(new AbstractAction("Generate NuSMV model") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.exportModelToNuSMV();
  }
}));
```

```
fileMenu.addSeparator();
fileMenu.add(new JMenuItem(new AbstractAction("Exit") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    System.exit(0);
  }
}));

add(fileMenu);

JMenu editMenu = new JMenu("Edit");

editMenu.add(new JMenuItem(new AbstractAction("Add new input") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.launchAddGraphElementDialog(Model.typeInput);
  }
}));

editMenu.add(new JMenuItem(new AbstractAction("Add new output") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.launchAddGraphElementDialog(Model.typeOutput);
  }
}));
editMenu.add(new JMenuItem(new AbstractAction("Add new signal") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.launchAddGraphElementDialog(Model.typeSignal);
  }
}));

editMenu.add(new JMenuItem(new AbstractAction("Add new register") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.launchAddGraphElementDialog(Model.typeRegister);
  }
}));

editMenu.addSeparator();

editMenu.add(new JMenuItem(new AbstractAction("Use module") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.launchAddGraphElementDialog(Model.typeModule);
  }
}));

editMenu.addSeparator();

editMenu.add(new JMenuItem(new AbstractAction("Add new state") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    app.launchAddGraphElementDialog(Model.typeState);
  }
}));

editMenu.add(new JMenuItem(new AbstractAction("Add new transition") {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
```

```
        app.launchAddGraphElementDialog(Model.typeTransition);
      }
    }));
    add(editMenu);

    JMenu viewMenu = new JMenu("View");

    viewMenu.add(new JMenuItem(
        new AbstractAction("Zoom in datapath graph") {
          static final long serialVersionUID = 1;

          public void actionPerformed(ActionEvent e) {
            app.setDatapathGraphScale(3, null);
          }
        }));

    viewMenu.add(new JMenuItem(
        new AbstractAction("Zoom out datapath graph") {
          static final long serialVersionUID = 1;

          public void actionPerformed(ActionEvent e) {
            app.setDatapathGraphScale(0.3, null);
          }
        }));

    viewMenu.addSeparator();

    viewMenu.add(new JMenuItem(new AbstractAction(
        "Zoom in controller graphl") {
      static final long serialVersionUID = 1;

      public void actionPerformed(ActionEvent e) {
        app.setFsmGraphScale(3, null);
      }
    }));

    viewMenu.add(new JMenuItem(new AbstractAction(
        "Zoom out controller graph") {
      static final long serialVersionUID = 1;

      public void actionPerformed(ActionEvent e) {
        app.setFsmGraphScale(0.3, null);
      }
    }));

    viewMenu.add(new JMenuItem(new AbstractAction("Collapse") {
      static final long serialVersionUID = 1;

      public void actionPerformed(ActionEvent e) {
        app.colpaseDatapathCells();
      }
    }));

    add(viewMenu);
  }
}
```

B.6.20 MaaohaToolBar.java

```
package maaoha.view;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.URL;
```

```
import javax.swing.AbstractAction;
import javax.swing.ImageIcon;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JToolBar;

import maaoha.model.Model;
import maaoha.view.model.DatapathView;
import maaoha.view.model.FsmTransitionView;
import maaoha.view.model.FsmView;
import maaoha.view.model.ModuleView;
import maaoha.view.model.SfgView;

/*
 * Class implements tool bar
 */
public class MaaohaToolBar extends JToolBar {
  static final long serialVersionUID = 1;

  private ToolView toolView;

  private JComboBox comboCurrentModule;

  private JComboBox comboViewType;

  private JComboBox comboSelectedView;

  private AbstractAction addModuleAction;

  private AbstractAction removeModuleAction;

  private AbstractAction addSfgAction;

  private AbstractAction editSfgAction;

  /*
   * Enables remove sfg button buttons
   */

  private AbstractAction removeSfgAction;

  private AbstractAction addUseModuleAction;

  private AbstractAction addRegisterAction;

  private AbstractAction addSignalAction;

  private AbstractAction addInputAction;

  private AbstractAction addOutputAction;

  private AbstractAction editControllerGraphAction;

  private AbstractAction deleteControllerGraphAction;

  private AbstractAction zoomControllerGraphAction;

  private AbstractAction zoominControllerGraphAction;

  private AbstractAction zoomoutControllerGraphAction;
  private AbstractAction editDatapathGraphAction;
  private AbstractAction deleteDatapathGraphAction;
  private AbstractAction zoomDatapathGraphAction;
  private AbstractAction zoominDatapathGraphAction;
  private AbstractAction zoomoutDatapathGraphAction;

  /*
   * Constructor
```

```
 */
public MaaohaToolBar(ToolView toolView) {
  this.toolView = toolView;
  createToolBar();
}

/*
 * Disables buttons when no module is present
 */
public void actionsDisableNoModules() {
  removeModuleAction.setEnabled(false);
  addSfgAction.setEnabled(false);
  editSfgAction.setEnabled(false);
  removeSfgAction.setEnabled(false);
  addUseModuleAction.setEnabled(false);
  addRegisterAction.setEnabled(false);
  addSignalAction.setEnabled(false);
  addInputAction.setEnabled(false);
  addOutputAction.setEnabled(false);
  editControllerGraphAction.setEnabled(false);
  deleteControllerGraphAction.setEnabled(false);
  zoomControllerGraphAction.setEnabled(false);
  zoominControllerGraphAction.setEnabled(false);
  zoomoutControllerGraphAction.setEnabled(false);
  editDatapathGraphAction.setEnabled(false);
  deleteDatapathGraphAction.setEnabled(false);
  zoomDatapathGraphAction.setEnabled(false);
  zoominDatapathGraphAction.setEnabled(false);
  zoomoutDatapathGraphAction.setEnabled(false);
  editSfgAction.setEnabled(false);
  removeSfgAction.setEnabled(false);
}

/*
 * Disables buttons when no sgf is present
 */
public void actionsDisableNoSfgs() {
  editSfgAction.setEnabled(false);
  removeSfgAction.setEnabled(false);
}

/*
 * Enables buttons when module is present
 */
public void actionsEnableModulePresent() {
  removeModuleAction.setEnabled(true);
  addSfgAction.setEnabled(true);
  addUseModuleAction.setEnabled(true);
  addRegisterAction.setEnabled(true);
  addSignalAction.setEnabled(true);
  addInputAction.setEnabled(true);
  addOutputAction.setEnabled(true);
  editControllerGraphAction.setEnabled(true);
  deleteControllerGraphAction.setEnabled(true);
  zoomControllerGraphAction.setEnabled(true);
  zoominControllerGraphAction.setEnabled(true);
  zoomoutControllerGraphAction.setEnabled(true);
  editDatapathGraphAction.setEnabled(true);
  deleteDatapathGraphAction.setEnabled(true);
  zoomDatapathGraphAction.setEnabled(true);
  zoominDatapathGraphAction.setEnabled(true);
  zoomoutDatapathGraphAction.setEnabled(true);
  removeSfgAction.setEnabled(true);
  editSfgAction.setEnabled(true);
}

/*
 * Enables buttons when sfg is present
```

```
 */
public void actionsEnableSfgPresent() {
  removeSfgAction.setEnabled(true);
  editSfgAction.setEnabled(true);
}

/*
 * Creates tool bar elements
 */
private void createToolBar() {
  setFloatable(false);
  URL insertUrl;
  ImageIcon insertIcon;
  insertUrl = getClass().getClassLoader().getResource(
      "resources/maaohaAddModule.gif");
  insertIcon = new ImageIcon(insertUrl);

  addModuleAction = new AbstractAction("", insertIcon) {
    static final long serialVersionUID = 1;

    public void actionPerformed(ActionEvent e) {
      System.out.println("razraz " + toolView.getModelView());
      AddModuleDialog cd = new AddModuleDialog(maaoha.MyTool.frame,
          toolView.getModelView(), null);
      cd.pack();
      cd.setVisible(true);
    }
  };
  add(addModuleAction);

  insertUrl = getClass().getClassLoader().getResource(
      "resources/maaohaRemoveModule.gif");
  insertIcon = new ImageIcon(insertUrl);
  removeModuleAction = new AbstractAction("", insertIcon) {
    static final long serialVersionUID = 1;

    public void actionPerformed(ActionEvent e) {
      RemoveModuleDialog cd = new RemoveModuleDialog(
          maaoha.MyTool.frame, toolView.getModelView(), null);
      cd.pack();
      cd.setVisible(true);
    }
  };
  add(removeModuleAction);

  insertUrl = getClass().getClassLoader().getResource(
      "resources/maaohaAddSfg.gif");
  insertIcon = new ImageIcon(insertUrl);
  addSfgAction = new AbstractAction("", insertIcon) {
    static final long serialVersionUID = 1;

    public void actionPerformed(ActionEvent e) {
      AddSfgDialog cd = new AddSfgDialog(maaoha.MyTool.frame,
          toolView.getCurrentModuleView(), null);
      cd.pack();
      cd.setVisible(true);
    }
  };
  add(addSfgAction);

  insertUrl = getClass().getClassLoader().getResource(
      "resources/maaohaEditSfg.gif");
  insertIcon = new ImageIcon(insertUrl);
  editSfgAction = new AbstractAction("", insertIcon) {
    static final long serialVersionUID = 1;

    public void actionPerformed(ActionEvent e) {
      EditSfgDialog cd = new EditSfgDialog(maaoha.MyTool.frame,
```

```
        toolView.getCurrentModuleView(), null);
    cd.pack();
    cd.setVisible(true);
  }
};
add(editSfgAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaRemoveSfg.gif");
insertIcon = new ImageIcon(insertUrl);
removeSfgAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    RemoveSfgDialog cd = new RemoveSfgDialog(maaoha.MyTool.frame,
        toolView.getCurrentModuleView(), null);
    cd.pack();
    cd.setVisible(true);
  }
};
add(removeSfgAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaAddUseModule.gif");
insertIcon = new ImageIcon(insertUrl);
addUseModuleAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    toolView.launchAddGraphElementDialog(Model.typeModule);
  }
};
add(addUseModuleAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaAddRegister.gif");
insertIcon = new ImageIcon(insertUrl);
addRegisterAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    toolView.launchAddGraphElementDialog(Model.typeRegister);
  }
};
add(addRegisterAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaAddSignal.gif");
insertIcon = new ImageIcon(insertUrl);
addSignalAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    toolView.launchAddGraphElementDialog(Model.typeSignal);
  }
};
add(addSignalAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaAddInput.gif");
insertIcon = new ImageIcon(insertUrl);
addInputAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    toolView.launchAddGraphElementDialog(Model.typeInput);
  }
};
```

```
add(addInputAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaAddOutput.gif");
insertIcon = new ImageIcon(insertUrl);
addOutputAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    toolView.launchAddGraphElementDialog(Model.typeOutput);
  }
};
add(addOutputAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaEdit.gif");
insertIcon = new ImageIcon(insertUrl);
editDatapathGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
  }
};
add(editDatapathGraphAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaDelete.gif");
insertIcon = new ImageIcon(insertUrl);
deleteDatapathGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
  }
};
add(deleteDatapathGraphAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/zoom.gif");
insertIcon = new ImageIcon(insertUrl);
zoomDatapathGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    toolView.setDatapathGraphScale(1, null);
  }
};
add(zoomDatapathGraphAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/zoomin.gif");
insertIcon = new ImageIcon(insertUrl);
zoominDatapathGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    double scale = toolView.getDatapathGraphScale();
    toolView.setDatapathGraphScale(scale * 1.1, null);
  }
};
add(zoominDatapathGraphAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/zoomout.gif");
insertIcon = new ImageIcon(insertUrl);
zoomoutDatapathGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
```

```
    double scale = toolView.getDatapathGraphScale();
    System.out.println("" + scale);
    toolView.setDatapathGraphScale(scale * 0.9, null);
  }
};
add(zoomoutDatapathGraphAction);

JLabel currentModule = new JLabel(" Module: ");
add(currentModule);
comboCurrentModule = new JComboBox();
updateCurrentModuleCombo();
comboCurrentModule.setMaximumSize(new Dimension(80, 60));
add(comboCurrentModule);

comboCurrentModule.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
  }
});

JLabel labelViewType = new JLabel(" View type: ");
add(labelViewType);
String viewTypes[] = { "Datapath", "SFG", "Transition" };
comboViewType = new JComboBox(viewTypes);
comboViewType.setMaximumSize(new Dimension(70, 60));
comboViewType.setEnabled(false);
add(comboViewType);

JLabel selectedViewLabel = new JLabel(" Selected: ");
add(selectedViewLabel);
comboSelectedView = new JComboBox();
updateSelectedViewCombo();
comboSelectedView.setMaximumSize(new Dimension(100, 60));
comboSelectedView.setEnabled(false);
add(comboSelectedView);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaEdit.gif");
insertIcon = new ImageIcon(insertUrl);
editControllerGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
  }
};
add(editControllerGraphAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/maaohaDelete.gif");
insertIcon = new ImageIcon(insertUrl);
deleteControllerGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
  }
};
add(deleteControllerGraphAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/zoom.gif");
insertIcon = new ImageIcon(insertUrl);
zoomControllerGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    toolView.setFsmGraphScale(1, null);
  }
};
add(zoomControllerGraphAction);
```

```
insertUrl = getClass().getClassLoader().getResource(
    "resources/zoomin.gif");
insertIcon = new ImageIcon(insertUrl);
zoominControllerGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    double scale = toolView.getFsmGraphScale();
    toolView.setFsmGraphScale(scale * 1.1, null);
  }
};
add(zoominControllerGraphAction);

insertUrl = getClass().getClassLoader().getResource(
    "resources/zoomout.gif");
insertIcon = new ImageIcon(insertUrl);
zoomoutControllerGraphAction = new AbstractAction("", insertIcon) {
  static final long serialVersionUID = 1;

  public void actionPerformed(ActionEvent e) {
    double scale = toolView.getFsmGraphScale();
    toolView.setFsmGraphScale(scale * 0.9, null);
  }
};
add(zoomoutControllerGraphAction);

comboCurrentModule.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    if (toolView.getListenersOn()) {
      toolView.setListenersOn(false);

      int index = comboCurrentModule.getSelectedIndex();
      if (index > -1) {
        comboViewType.setEnabled(true);
        comboViewType.setSelectedIndex(0);
        toolView.setVisibleModule(toolView.getModelView()
            .getModuleView(index).getView());
        toolView.getModelView().getModuleView(index)
            .getDatapath().setCurrentViewDatapath();
      }
    }
    toolView.setListenersOn(true);
  }
});

comboViewType.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    updateSelectedViewCombo();
  }
});

comboSelectedView.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {

    if (toolView.getListenersOn()) {
      toolView.setListenersOn(false);
      int moduleIndex = comboCurrentModule.getSelectedIndex();
      int modeIndex = comboViewType.getSelectedIndex();
      int selectedIndex = comboSelectedView.getSelectedIndex();
      if (selectedIndex > -1
          && selectedIndex < comboSelectedView.getItemCount()) {
        if (modeIndex == 0) {
          DatapathView dv = toolView.getModelView()
              .getModuleView(moduleIndex).getDatapath();
          dv.setCurrentViewDatapath();
        } else if (modeIndex == 1) {
          DatapathView dv = toolView.getModelView()
```

```
                .getModuleView(moduleIndex).getDatapath();
            dv.setCurrentViewSfg(dv.getSfg(selectedIndex));
          } else if (modeIndex == 2) {
            ModuleView mv = toolView.getModelView()
                .getModuleView(moduleIndex);
            mv.getDatapath().setCurrentViewTransition(
                mv.getFsmV().getFsmTransitionByTotalIndex(
                    selectedIndex));
          }
        }
      }
      toolView.setListenersOn(true);
    }
  });
}

/*
 * Enables add input button buttons
 */
public void setAddInputActionEnabled(boolean b) {
  addInputAction.setEnabled(b);
}

/*
 * Enables add module button buttons
 */
public void setAddModuleActionEnabled(boolean b) {
  addModuleAction.setEnabled(b);
}

/*
 * Enables add output button buttons
 */
public void setAddOutputActionnEnabled(boolean b) {
  addOutputAction.setEnabled(b);
}

/*
 * Enables add register button buttons
 */
public void setAddRegisterActionEnabled(boolean b) {
  addRegisterAction.setEnabled(b);
}

/*
 * Enables add signal button buttons
 */
public void setAddSignalActionEnabled(boolean b) {
  addSignalAction.setEnabled(b);
}

/*
 * Enables add sub-module button buttons
 */
public void setAddUseModuleActionEnabled(boolean b) {
  addUseModuleAction.setEnabled(b);
}

/*
 * Sets module combo
 */
public void setComboCurrentModule(Component view) {
  int index = toolView.getModelView().getIndexOfModuleViewByView(view);
  if (index > -1 && index < comboCurrentModule.getItemCount())
    comboCurrentModule.setSelectedIndex(index);
}

/*
```

```
 * Enables edit sfg button buttons
 */
public void setEditSfgActionEnabled(boolean b) {
  editSfgAction.setEnabled(b);
}

/*
 * Disables add module button buttons
 */
public void setRemoveModuleActionEnabled(boolean b) {
  removeModuleAction.setEnabled(b);
}

public void setRemoveSfgActionEnabled(boolean b) {
  removeSfgAction.setEnabled(b);
}

/*
 * Sets view type combo
 */
public void setViewTypeCombo(Object obj) {
  if (obj instanceof DatapathView) {
    comboViewType.setSelectedIndex(0);
  } else if (obj instanceof SfgView) {
    SfgView sfgView = (SfgView) obj;
    comboViewType.setSelectedIndex(1);
    int index = sfgView.getParent().getSfgIndex(sfgView);
    if (index > -1 & index < comboSelectedView.getItemCount())
      comboSelectedView.setSelectedIndex(index);
  } else if (obj instanceof FsmTransitionView) {
    FsmTransitionView transView = (FsmTransitionView) obj;
    comboViewType.setSelectedIndex(2);
    int index = transView.getParent().getParent()
        .getTransitionViewIndex(transView);
    if (index > -1 & index < comboSelectedView.getItemCount())
      comboSelectedView.setSelectedIndex(index);
  }
}

/*
 * Updates module combo
 */
public void updateCurrentModuleCombo() {
  comboCurrentModule.removeAllItems();
  for (int i = 0; i < toolView.getModelView().getModulesNo(); i++) {
    comboCurrentModule.addItem(toolView.getModelView().getModuleView(i)
        .getModule().getName());
  }
  if (toolView.getModelView().getModulesNo() > 0)
    comboViewType.setEnabled(true);
}

/*
 * Updates view combo
 */
public void updateSelectedViewCombo() {

  comboSelectedView.removeAllItems();
  int icomboViewType = comboViewType.getSelectedIndex();
  if (icomboViewType == 0) {
    comboSelectedView.setEnabled(false);
  } else if (icomboViewType == 1) {
    ModuleView moduleView = toolView.getModelView().getModuleView(
        comboCurrentModule.getSelectedIndex());
    for (int i = 0; i < moduleView.getDatapath().getSfgsNo(); i++) {
      comboSelectedView.addItem(moduleView.getDatapath().getSfg(i));
    }
    comboSelectedView.setEnabled(true);
```

```
    } else if (icomboViewType == 2) {
      FsmView fsmView = toolView.getModelView().getModuleView(
          comboCurrentModule.getSelectedIndex()).getFsmV();
      for (int i = 0; i < fsmView.getFsmStateNo(); i++) {
        for (int j = 0; j < fsmView.getFsmState(i)
            .getTransitionViewNo(); j++) {
          FsmTransitionView transV = fsmView.getFsmState(i)
              .getTransitionView(j);
          comboSelectedView.addItem(fsmView.getFsmState(i)
              .getFsmState().getName()
              + " : "
              + (j + 1)
              + ". -> "
              + transV.getTransition().getNextState().getName());
        }
      }
      comboSelectedView.setEnabled(true);
    }
  }
}
```

B.6.21 RemoveModuleDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;

import maaoha.MaaohaException;
import maaoha.view.model.ModelView;
import maaoha.view.model.ModuleView;

/*
 * Class implements dialog for removal of modules
 */
public class RemoveModuleDialog extends JDialog {
  static final long serialVersionUID = 1;

  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public RemoveModuleDialog(Frame aFrame, final ModelView model, Point point) {
    super(aFrame, true);
    if (point != null)
      setLocation(point.x, point.y);
    setTitle("Remove module...");

    final String msgString0 = "old name:";
    final JComboBox moduleCombo = new JComboBox();

    for (int i = 0; i < model.getModulesNo(); i++) {
      ModuleView moduleView = model.getModuleView(i);
      moduleCombo.addItem(moduleView.getModule().getName());
    }

    Object[] array = { msgString0, moduleCombo };
    final String btnString1 = "Remove";
```

```
    final String btnString2 = "Cancel";

    Object[] options = { btnString1, btnString2 };

    optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
        JOptionPane.YES_NO_OPTION, null, options, options[0]);
    setContentPane(optionPane);
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
      }
    });

    optionPane.addPropertyChangeListener(new PropertyChangeListener() {
      public void propertyChange(PropertyChangeEvent e) {
        String prop = e.getPropertyName();

        if (isVisible()
            && (e.getSource() == optionPane)
            && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
                .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
          Object value = optionPane.getValue();

          if (value == JOptionPane.UNINITIALIZED_VALUE) {
            // ignore reset
            return;
          }

          optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

          if (value.equals(btnString1)) {
            try {
              model.removeModule(model.getModuleView(moduleCombo
                  .getSelectedIndex()));
              if (model.getModulesNo() == 0) {
                model.getToolView().getToolBar()
                    .actionsDisableNoModules();
              }
            } catch (MaaohaException ex) {
              // text was invalid
              JOptionPane.showMessageDialog(
                  RemoveModuleDialog.this, "Problems!!!:\n",
                  "Try again", JOptionPane.ERROR_MESSAGE);
            }
            setVisible(false);
          } else {
            setVisible(false);
          }
        }
      }
    });
  }
}
```

B.6.22 RemoveSfgDialog.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JComboBox;
```

```
import javax.swing.JDialog;
import javax.swing.JOptionPane;

import maaoha.view.model.ModuleView;
import maaoha.view.model.SfgView;

/*
 * Class implements dialog for removal of signal flow graphs
 */
public class RemoveSfgDialog extends JDialog {
  static final long serialVersionUID = 1;

  private JOptionPane optionPane;

  /*
   * Constructor
   */
  public RemoveSfgDialog(Frame aFrame, final ModuleView module, Point point) {
    super(aFrame, true);

    if (point != null)
      setLocation(point.x, point.y);
    setTitle("Remove signal flow graph...");

    final String msgString0 = "old name:";
    final JComboBox sfgCombo = new JComboBox();

    for (int i = 0; i < module.getModule().getDatapath().getSfgsNo(); i++) {
      SfgView sfg = module.getDatapath().getSfg(i);
      sfgCombo.addItem(sfg);
    }

    Object[] array = { msgString0, sfgCombo };

    final String btnString1 = "Remove";
    final String btnString2 = "Cancel";

    Object[] options = { btnString1, btnString2 };

    optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
        JOptionPane.YES_NO_OPTION, null, options, options[0]);
    setContentPane(optionPane);
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
      }
    });

    optionPane.addPropertyChangeListener(new PropertyChangeListener() {
      public void propertyChange(PropertyChangeEvent e) {
        String prop = e.getPropertyName();

        if (isVisible()
            && (e.getSource() == optionPane)
            && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
                .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
          Object value = optionPane.getValue();

          if (value == JOptionPane.UNINITIALIZED_VALUE) {
            // ignore reset
            return;
          }

          optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

          if (value.equals(btnString1)) {
            module.removeSfg(module.getDatapath().getSfg(
```

```
                sfgCombo.getSelectedIndex()));
            if (module.getDatapath().getSfgsNo() == 0) {
              module.getParent().getToolView().getToolBar()
                  .actionsDisableNoSfgs();
            }
            setVisible(false);
          } else {
            setVisible(false);
          }
        }
      }
    });
  }
}
```

B.6.23 SyncTreeSelectionListener.java

```
package maaoha.view;

import javax.swing.event.TreeSelectionEvent;
import javax.swing.event.TreeSelectionListener;
import javax.swing.tree.DefaultMutableTreeNode;

import maaoha.view.model.DatapathView;
import maaoha.view.model.FsmStateView;
import maaoha.view.model.FsmTransitionView;
import maaoha.view.model.FsmView;
import maaoha.view.model.ModuleView;
import maaoha.view.model.SfgView;

/*
 * Class implements tree selecetion listener
 */
public class SyncTreeSelectionListener implements TreeSelectionListener {
  protected ToolView parent;

  /*
   * Constructor
   */
  public SyncTreeSelectionListener(ToolView parent) {
    this.parent = parent;
  }

  /*
   * (non-Javadoc)
   *
   * @see
javax.swing.event.TreeSelectionListener#valueChanged(javax.swing.event.TreeSelectio
nEvent)
   */
  public void valueChanged(TreeSelectionEvent e) {
    DefaultMutableTreeNode node = (DefaultMutableTreeNode) parent.getTree()
        .getLastSelectedPathComponent();

    // if nothing is selected
    if (node == null)
      return;

    // retrieve the node that was selected
    Object userObject = node.getUserObject();

    if (userObject instanceof TreeInfo) {
      if (parent.getListenersOn()) {
        parent.setListenersOn(false);
        Object modelObject = ((TreeInfo) userObject).getObject();
        if (modelObject instanceof SfgView) {
          SfgView sfgView = (SfgView) modelObject;
```

```
            DatapathView datapathView = sfgView.getParent();
            datapathView.setCurrentViewSfg(sfgView);
            parent.getToolBar().setViewTypeCombo(sfgView);
          } else if (modelObject instanceof DatapathView) {
            // displays proper module, sets view to datapath elements
            // view
            DatapathView datapathView = (DatapathView) modelObject;
            datapathView.setCurrentViewDatapath();
            parent.getToolBar().setViewTypeCombo(datapathView);
          } else if (modelObject instanceof ModuleView) {
            // displays proper module
            ModuleView module = (ModuleView) modelObject;
            parent.modulesPane.setSelectedComponent(module.getView());
          } else if (modelObject instanceof FsmTransitionView) {
            // displays proper module, sets view to transition view
            FsmTransitionView trans = (FsmTransitionView) modelObject;

            DatapathView datapathView = trans.getParent().getParent()
                .getParent().getDatapath();
            datapathView.setCurrentViewTransition(trans);
            parent.getToolBar().setViewTypeCombo(trans);
          } else if (modelObject instanceof FsmStateView) {
            // displays proper module, sets view to transition view
            // (with toggling between transitions)
            FsmStateView stateView = (FsmStateView) modelObject;
            parent.modulesPane.setSelectedComponent(stateView
                .getParent().getParent().getView());
          } else if (modelObject instanceof FsmView) {
            // displays proper module
            FsmView fsmView = (FsmView) modelObject;
            parent.modulesPane.setSelectedComponent(fsmView.getParent()
                .getView());
          }
        }
        parent.setListenersOn(true);
      }
    }
}
```

B.6.24 ToolView.java

```
package maaoha.view;

import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.geom.Point2D;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

import javax.swing.JApplet;
import javax.swing.JFileChooser;
import javax.swing.JScrollPane;
import javax.swing.JSplitPane;
import javax.swing.JTabbedPane;
import javax.swing.JTree;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.TreeSelectionModel;

import maaoha.MaaohaException;
import maaoha.model.Model;
```

```
import maaoha.model.*;
import maaoha.model.expression.*;

import maaoha.source.GezelGenerator;
import maaoha.source.GezelParser;
import maaoha.source.NuSMVGenerator;
import maaoha.view.model.ModelView;
import maaoha.view.model.ModuleView;
import maaoha.view.serialization.ModelData;

import com.thoughtworks.xstream.XStream;

/*
 * Main view of the tool
 */
public class ToolView extends JApplet {
  static final long serialVersionUID = 1;

  protected boolean listenersOn;

  protected JScrollPane treePane;

  protected JTree tree;

  protected Model model;

  protected ModelView modelView;

  protected JSplitPane graphsPane;

  protected JTabbedPane modulesPane;

  protected DefaultTreeModel treeModel;

  protected boolean displayIndexOnEdge;

  protected boolean displayConditionOnEdge;

  protected boolean displaySfgsOnEdge;

  protected boolean verifyModel;

  protected JFileChooser fileChooser;

  protected MaaohaToolBar toolBar;

  protected boolean ignoreEvents;

  /*
   * Constructor
   */
  public ToolView() {
    displayIndexOnEdge = true;
    displayConditionOnEdge = true;
    displaySfgsOnEdge = true;
    verifyModel = true;
    ignoreEvents = false;

    // initialize model and model view
    try {
      model = new Model();
      modelView = new ModelView(model, this);
      populateMainPane();
      toolBar.actionsDisableNoModules();
    } catch (MaaohaException e) {
      System.out.println(e.getMessage());
      System.exit(1);
    }
```

```
  fileChooser = new JFileChooser();
}

/*
 * Populates module pane with module view
 */
public void addModuleViewToModulesPane(ModuleView moduleView)
    throws MaaohaException {
  JSplitPane graphsPane = moduleView.createView();
  graphsPane.setDividerLocation(400);
  modulesPane.addTab(moduleView.getModule().getName(), graphsPane);
}

/*
 * Collapes datapath graph cells
 */
public void colpaseDatapathCells() {
  modelView.colapseDatapathCells(modulesPane.getSelectedComponent());
}

/*
 * Resets current model
 */
public void createNewModel() {
  try {
    model = new Model();
    modelView = new ModelView(model, this);
    treeModel = new DefaultTreeModel(modelView.createTree());
    tree.setModel(treeModel);
    ignoreEvents = true;
    modulesPane.removeAll();
    ignoreEvents = false;
    modelView.populateModulesPane(modulesPane);
    toolBar.actionsDisableNoModules();
  } catch (MaaohaException ex) {
    System.out.println(ex);
  }
}

/*
 * Generates NuSMV program
 */
public void exportModelToNuSMV() {
  int returnVal = fileChooser.showOpenDialog(this);
  if (returnVal == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();

    try {
      BufferedWriter out = new BufferedWriter(new FileWriter(file));
      out.write(NuSMVGenerator.GenerateNuSMVModel(model));
      out.close();
    } catch (IOException e) {
      System.out.print(e.toString());
    } catch (MaaohaException e) {
      System.out.print(e.toString());
    }
  }
}

/*
 * Returns current module view
 */
public ModuleView getCurrentModuleView() {
  return modelView.findModuleViewBySelectedComponent(modulesPane
      .getSelectedComponent());
}
```

```
/*
 * Returns datapath graph scale
 */
public double getDatapathGraphScale() {
  return modelView.getDatapathGraphScale(modulesPane
      .getSelectedComponent());
}

/*
 * Checks wheteher condition should appear on the controller graph labels
 */
public boolean getDisplayConditionOnEdge() {
  return displayConditionOnEdge;
}

/*
 * Checks wheteher transition index should appear on the controller graph
 * labels
 */
public boolean getDisplayIndexOnEdge() {
  return displayIndexOnEdge;
}

/*
 * Checks wheteher list of sfgs should appear on the controller graph labels
 */
public boolean getDisplaySfgsOnEdge() {
  return displaySfgsOnEdge;
}

/*
 * Returns controller graph scale
 */
public double getFsmGraphScale() {
  return modelView.getFsmGraphScale(modulesPane.getSelectedComponent());
}

/*
 * Checks wheteher listeners are activated
 */
public boolean getListenersOn() {
  return listenersOn;
}

/*
 * Returns model
 */
public Model getModel() {
  return model;
}

/*
 * Returns module view
 */
public ModelView getModelView() {
  return modelView;
}

/*
 * Returns tool bar
 */
public MaaohaToolBar getToolBar() {
  return toolBar;
}

/*
 * Returns tree object
 */
```

```
public JTree getTree() {
  return tree;
}

/*
 * Returns tree model
 */
public DefaultTreeModel getTreeModel() {
  return treeModel;
}

/*
 * Checks whetеher model should be verified
 */
public boolean getVerifyModel() {
  return verifyModel;
}

/*
 * Invokes apprioprate dialog for element addition
 */
public void launchAddGraphElementDialog(String type) {
  try {
    modelView.addNewGraphElement(modulesPane.getSelectedComponent(),
        type);
  } catch (MaaohaException me) {
    System.out.println(me.toString());
  }
}

/*
 * Populates main pane
 */
protected void populateMainPane() throws MaaohaException {
  treeModel = new DefaultTreeModel(modelView.createTree());
  tree = new JTree(treeModel);
  treePane = new JScrollPane(tree);
  tree.getSelectionModel().setSelectionMode(
      TreeSelectionModel.SINGLE_TREE_SELECTION);
  tree.addTreeSelectionListener(new SyncTreeSelectionListener(this));

  modulesPane = new JTabbedPane();
  modulesPane.addChangeListener(new ChangeListener() {
    public void stateChanged(ChangeEvent event) {
      if (!ignoreEvents) {
        toolBar.setComboCurrentModule(modulesPane
            .getSelectedComponent());
        if (modulesPane.getSelectedComponent() != null)
          modelView.refreshTabbedPane(modulesPane
              .getSelectedComponent());
      }
    }
  });
  modelView.populateModulesPane(modulesPane);

  JSplitPane mainPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
      treePane, modulesPane);
  mainPane.setDividerLocation(220);
  getContentPane().add(mainPane, BorderLayout.CENTER);

  toolBar = new MaaohaToolBar(this);
  setJMenuBar(new MaaohaMenuBar(this));
  getContentPane().add(toolBar, BorderLayout.NORTH);
}

/*
 * Imports model from GEZEL program
 */
```

```
public void readModelFromGezelSource() {
  int returnVal = fileChooser.showOpenDialog(this);
  if (returnVal == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();
    try {
      BufferedReader in = new BufferedReader(new FileReader(file));
      GezelParser gp = new GezelParser();
      model = new Model();
      gp.parseGezelCode(in, model);
      in.close();
      modelView = new ModelView(model, this);
      treeModel = new DefaultTreeModel(modelView.createTree());
      tree.setModel(treeModel);
      ignoreEvents = true;
      modulesPane.removeAll();
      ignoreEvents = false;
      modelView.populateModulesPane(modulesPane);
      modelView.checkModel();
      toolBar.updateCurrentModuleCombo();
      if (model.getModulesNo() == 0)
        toolBar.actionsDisableNoModules();
      else
        toolBar.actionsEnableModulePresent();
    } catch (IOException e) {
      System.out.println(e);
    } catch (MaaohaException ex) {
      System.out.println(ex);
    }
  }
}

/*
 * Imports model from XML file
 */
public void readModelFromXML() {
  XStream xstream = new XStream();

  xstream.alias("model", Model.class);
  xstream.alias("module", Module.class);
  xstream.alias("element", DatapathElement.class);
  xstream.alias("useModule", UseModule.class);
  xstream.alias("expression", Expression.class);
  xstream.alias("condition", Condition.class);
  xstream.alias("basicExpression", BasicExpression.class);
  xstream.alias("binaryOperator", BinaryOperator.class);
  xstream.alias("bitSelection", BitSelection.class);
  xstream.alias("constant", Constant.class);
  xstream.alias("lookupTableOperator", LookupTableOperator.class);
  xstream.alias("ternaryOperator", TernaryOperator.class);
  xstream.alias("UnaryOperator", UnaryOperator.class);
  xstream.alias("variable", Variable.class);
  xstream.alias("sfg", Sfg.class);
  xstream.alias("datapath", Datapath.class);
  xstream.alias("fsm", Fsm.class);
  xstream.alias("fsmState", FsmState.class);
  xstream.alias("fsmTransition", FsmTransition.class);
  xstream.alias("lookupTable", LookUpTable.class);
  int returnVal = fileChooser.showOpenDialog(this);
  if (returnVal == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();
    try {
      BufferedReader in = new BufferedReader(new FileReader(file));
      model = (Model) xstream.fromXML(in);
      in.close();
      modelView = new ModelView(model, this);
      treeModel = new DefaultTreeModel(modelView.createTree());
      tree.setModel(treeModel);
      ignoreEvents = true;
```

```
      modulesPane.removeAll();
      ignoreEvents = false;
      modelView.populateModulesPane(modulesPane);
      modelView.checkModel();
      toolBar.updateCurrentModuleCombo();
      if (model.getModulesNo() == 0)
        toolBar.actionsDisableNoModules();
      else
        toolBar.actionsEnableModulePresent();
    } catch (IOException e) {
      System.out.print(e.toString());
    } catch (MaaohaException ex) {
      System.out.println(ex);
    }
  }
}

/*
 * Imports project from XML file
 */
public void readModelViewFromXML() {
  XStream xstream = new XStream();
  xstream.alias("model", Model.class);
  xstream.alias("module", Module.class);
  xstream.alias("element", DatapathElement.class);
  xstream.alias("useModule", UseModule.class);
  xstream.alias("expression", Expression.class);
  xstream.alias("condition", Condition.class);
  xstream.alias("basicExpression", BasicExpression.class);
  xstream.alias("binaryOperator", BinaryOperator.class);
  xstream.alias("bitSelection", BitSelection.class);
  xstream.alias("constant", Constant.class);
  xstream.alias("lookupTableOperator", LookupTableOperator.class);
  xstream.alias("ternaryOperator", TernaryOperator.class);
  xstream.alias("UnaryOperator", UnaryOperator.class);
  xstream.alias("variable", Variable.class);
  xstream.alias("sfg", Sfg.class);
  xstream.alias("datapath", Datapath.class);
  xstream.alias("fsm", Fsm.class);
  xstream.alias("fsmState", FsmState.class);
  xstream.alias("fsmTransition", FsmTransition.class);
  xstream.alias("lookupTable", LookUpTable.class);
  int returnVal = fileChooser.showOpenDialog(this);
  if (returnVal == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();
    try {
      BufferedReader in = new BufferedReader(new FileReader(file));
      ModelData modelData = (ModelData) xstream.fromXML(in);
      in.close();
      model = modelData.getModel();
      modelView = new ModelView(model, this);
      treeModel = new DefaultTreeModel(modelView.createTree());
      tree.setModel(treeModel);
      modulesPane.removeAll();
      modelView.populateModulesPane(modulesPane);
      modelData.restoreGraphCellPosition(modelView);
      if (model.getModulesNo() == 0)
        toolBar.actionsDisableNoModules();
      else
        toolBar.actionsEnableModulePresent();
    } catch (IOException e) {
      System.out.print(e.toString());
    } catch (MaaohaException ex) {
      System.out.println(ex);
    }
  }
}
```

```
  /*
   * Clear module panes
   */
  public void removeModuleViewFromModulesPane(ModuleView moduleView)
      throws MaaohaException {
    modulesPane.remove(moduleView.getView());//
.addTab(moduleView.getModule().getName(),
                          // graphsPane);
  }

  /*
   * Sets datapath graph scale
   */
  public void setDatapathGraphScale(double scale, Point2D.Double pt) {
    modelView.setDatapathGraphScale(modulesPane.getSelectedComponent(),
        scale, pt);
  }

  /*
   * Sets controller graph scale
   */
  public void setFsmGraphScale(double scale, Point2D.Double pt) {
    modelView.setFsmGraphScale(modulesPane.getSelectedComponent(), scale,
        pt);
  }

  /*
   * Sets wheteher listeners are activated
   */
  public void setListenersOn(boolean b) {
    listenersOn = b;
  }

  /*
   * Sets model
   */
  public void setModel(Model model) {
    this.model = model;
  }

  /*
   * Sets visible module
   */
  public void setVisibleModule(Component c) {
    if (c != null)
      modulesPane.setSelectedComponent(c);
  }

  /*
   * Generates GEZEL program
   */
  public void writeModelToGezelSource() {
    int returnVal = fileChooser.showOpenDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
      File file = fileChooser.getSelectedFile();

      try {
        BufferedWriter out = new BufferedWriter(new FileWriter(file));
        out.write(GezelGenerator.generateGezelCode(model));
        out.close();
      } catch (IOException e) {
        System.out.print(e.toString());
      }
    }
  }

  /*
   * Exports model to XML file
```

```
 */
public void writeModelToXML() {
  XStream xstream = new XStream();
  xstream.alias("model", Model.class);
  xstream.alias("module", Module.class);
  xstream.alias("element", DatapathElement.class);
  xstream.alias("useModule", UseModule.class);
  xstream.alias("expression", Expression.class);
  xstream.alias("condition", Condition.class);
  xstream.alias("basicExpression", BasicExpression.class);
  xstream.alias("binaryOperator", BinaryOperator.class);
  xstream.alias("bitSelection", BitSelection.class);
  xstream.alias("constant", Constant.class);
  xstream.alias("lookupTableOperator", LookupTableOperator.class);
  xstream.alias("ternaryOperator", TernaryOperator.class);
  xstream.alias("UnaryOperator", UnaryOperator.class);
  xstream.alias("variable", Variable.class);
  xstream.alias("sfg", Sfg.class);
  xstream.alias("datapath", Datapath.class);
  xstream.alias("fsm", Fsm.class);
  xstream.alias("fsmState", FsmState.class);
  xstream.alias("fsmTransition", FsmTransition.class);
  xstream.alias("lookupTable", LookUpTable.class);
  String xml = xstream.toXML(model);

  int returnVal = fileChooser.showOpenDialog(this);
  if (returnVal == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();

    try {
      BufferedWriter out = new BufferedWriter(new FileWriter(file));
      out.write(xml);
      out.close();
    } catch (IOException e) {
      System.out.print(e.toString());
    }
  }
}

/*
 * Exports project to XML file
 */
public void writeModelViewToXML() {
  XStream xstream = new XStream();
  xstream.alias("model", Model.class);
  xstream.alias("module", Module.class);
  xstream.alias("element", DatapathElement.class);
  xstream.alias("useModule", UseModule.class);
  xstream.alias("expression", Expression.class);
  xstream.alias("condition", Condition.class);
  xstream.alias("basicExpression", BasicExpression.class);
  xstream.alias("binaryOperator", BinaryOperator.class);
  xstream.alias("bitSelection", BitSelection.class);
  xstream.alias("constant", Constant.class);
  xstream.alias("lookupTableOperator", LookupTableOperator.class);
  xstream.alias("ternaryOperator", TernaryOperator.class);
  xstream.alias("UnaryOperator", UnaryOperator.class);
  xstream.alias("variable", Variable.class);
  xstream.alias("sfg", Sfg.class);
  xstream.alias("datapath", Datapath.class);
  xstream.alias("fsm", Fsm.class);
  xstream.alias("fsmState", FsmState.class);
  xstream.alias("fsmTransition", FsmTransition.class);
  xstream.alias("lookupTable", LookUpTable.class);
  ModelData modelData = new ModelData(modelView);
  String xml = xstream.toXML(modelData);

  int returnVal = fileChooser.showOpenDialog(this);
```

```
    if (returnVal == JFileChooser.APPROVE_OPTION) {
      File file = fileChooser.getSelectedFile();

      try {
        BufferedWriter out = new BufferedWriter(new FileWriter(file));
        out.write(xml);
        out.close();
      } catch (IOException e) {
        System.out.print(e.toString());
      }
    }
  }
}
```

B.6.25 TreeInfo.java

```
package maaoha.view;

/*
 * Auxiliary structure for storage of label
 * together with pointer to the object
 * being represented by the tree node
 */
public class TreeInfo {
  protected Object pointer;

  protected String label;

  /*
   * Constructor
   */
  public TreeInfo(String label, Object pointer) {
    this.label = label;
    this.pointer = pointer;
  }

  /*
   * Returns object
   */
  public Object getObject() {
    return pointer;
  }

  /*
   * (non-Javadoc)
   *
   * @see java.lang.Object#toString()
   */
  public String toString() {
    return label;
  }
}
```

B.6.26 TreeInfo.java

```
package maaoha.view;

import java.awt.Frame;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
```

```
import java.util.ArrayList;

import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import maaoha.MaaohaException;
import maaoha.model.DatapathElement;
import maaoha.model.Module;
import maaoha.tools.MaaohaTools;
import maaoha.view.model.ModuleView;

/*
 * Class implements dialog for usage of sub-module
 */
public class UseModuleDialog extends JDialog {
  static final long serialVersionUID = 1;

  private Module selectedModule = null;

  private String parametersText = null;
  private String nameText = null;
  private JOptionPane optionPane;
  private JTextField parametersField = null;
  private JTextField nameField = null;

  /*
   * Constructor
   */
  public UseModuleDialog(Frame aFrame, final ModuleView moduleView,
      final Point point) {
    super(aFrame, true);
    if (point != null)
      setLocation(point.x, point.y);
    setTitle("Use module...");

    final String msgString0 = "Module name:";
    nameField = new JTextField(30);
    final String msgString1 = "Use module:";
    final JComboBox moduleCombo = new JComboBox();
    final String msgString2 = "Parameters:";
    parametersField = new JTextField(30);

    int index = 0;

    for (int i = 0; i < moduleView.getParent().getModel().getModulesNo(); i++) {
      Module module = moduleView.getParent().getModel().getModule(i);
      moduleCombo.addItem(module.getName());
    }

    moduleCombo.setSelectedIndex(index);

    ArrayList<DatapathElement> de = moduleView.getParent().getModel()
        .getModule(moduleCombo.getSelectedIndex()).getDatapath()
        .getElements();
    parametersField.setText(generateParametersText(de));

    Object[] array = { msgString0, nameField, msgString1, moduleCombo,
        msgString2, parametersField };

    final String btnString1 = "Create";
    final String btnString2 = "Cancel";

    Object[] options = { btnString1, btnString2 };

    optionPane = new JOptionPane(array, JOptionPane.QUESTION_MESSAGE,
        JOptionPane.YES_NO_OPTION, null, options, options[0]);
```

```
setContentPane(optionPane);
setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent we) {
    optionPane.setValue(new Integer(JOptionPane.CLOSED_OPTION));
  }
});

moduleCombo.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    ArrayList<DatapathElement> de = moduleView.getParent()
        .getModel().getModule(moduleCombo.getSelectedIndex())
        .getDatapath().getElements();
    parametersField.setText(generateParametersText(de));
  }
});

parametersField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

nameField.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    optionPane.setValue(btnString1);
  }
});

optionPane.addPropertyChangeListener(new PropertyChangeListener() {
  public void propertyChange(PropertyChangeEvent e) {
    String prop = e.getPropertyName();

    if (isVisible()
        && (e.getSource() == optionPane)
        && (prop.equals(JOptionPane.VALUE_PROPERTY) || prop
            .equals(JOptionPane.INPUT_VALUE_PROPERTY))) {
      Object value = optionPane.getValue();

      if (value == JOptionPane.UNINITIALIZED_VALUE) {
        // ignore reset
        return;
      }

      optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

      if (value.equals(btnString1)) {
        selectedModule = moduleView.getParent().getModel()
            .getModule(moduleCombo.getSelectedIndex());
        parametersText = parametersField.getText();
        nameText = nameField.getText();

        if (!MaaohaTools.validName(nameText)) {
          // text was invalid
          nameField.selectAll();
          JOptionPane.showMessageDialog(UseModuleDialog.this,
              "Sorry, \"" + nameText + "\" "
                  + "is not a valid name.\n"
                  + "Please enter a different name.",
              "Try again", JOptionPane.ERROR_MESSAGE);
          nameText = null;
        } else if (moduleView.getModule().getDatapath()
            .elementExists(nameText)) {
          // name already in use
          nameField.selectAll();
          JOptionPane.showMessageDialog(UseModuleDialog.this,
              "Sorry, \"" + nameText + "\" "
                  + "already in use.\n"
```

```
                    + "Please enter a different name.",
                "Try again", JOptionPane.ERROR_MESSAGE);
            nameText = null;
          } else {
            // we're done; dismiss the dialog
            try {
              moduleView.getDatapath().createUseModule(point,
                  nameText, selectedModule,
                  parametersText);
              if (moduleView.getParent().getToolView()
                  .getVerifyModel()) {
                try {
                  moduleView.getParent().checkModel();
                } catch (MaaohaException ex) {
                  ex.toString();
                }
              }
            } catch (MaaohaException ex) {
              // invalid parameters
              JOptionPane.showMessageDialog(
                  UseModuleDialog.this,
                  "Invalid parameters:\n"
                      + parametersText, "Try again",
                  JOptionPane.ERROR_MESSAGE);
              parametersText = null;
              nameText = null;
            }
            setVisible(false);
          }
        } else {
          setVisible(false);
        }
      }
    }
  });
}

private String generateParametersText(ArrayList<DatapathElement> de) {
  String str = "";

  int in = 1;
  int out = 1;
  for (int i = 0; i < de.size(); i++) {
    String type = de.get(i).getElementType();
    if (type.equals(maaoha.model.Model.typeOutput)) {
      if (in + out > 2)
        str += " , ";
      str += type + out++;
    } else if (type.equals(maaoha.model.Model.typeInput)) {
      if (in + out > 2)
        str += " , ";
      str += type + in++;
    }
  }
  return str;
}
}
```

B.7 Package maaoha.view.model

B.7.1 DatapathElementView.java

```
package maaoha.view.model;

import javax.swing.tree.DefaultMutableTreeNode;
```

```
import maaoha.model.DatapathElement;
import maaoha.view.TreeInfo;

import org.jgraph.graph.DefaultGraphCell;

/*
 * Class responsible for visualization of datapath elements
 */
public class DatapathElementView implements GraphCellView {
  protected DefaultMutableTreeNode treeNode;

  protected DatapathElement element;

  protected DefaultGraphCell cell;

  protected DatapathView parentDatapathView;

  /*
   * Constructor
   */
  public DatapathElementView(DatapathElement element,
      DatapathView parentDatapathView) {
    this.element = element;
    this.parentDatapathView = parentDatapathView;
  }

  /*
   * Creates tree node
   */
  public DefaultMutableTreeNode createTree() {
    treeNode = new DefaultMutableTreeNode(new TreeInfo(element.getName()
        + " : " + element.getValueType(), this));
    return treeNode;
  }

  /*
   * Returns element
   */
  public DatapathElement getElement() {
    return element;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.view.model.GraphCellView#getGraphCell()
   */
  public DefaultGraphCell getGraphCell() {
    return cell;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.view.model.GraphCellView#getModelObject()
   */
  public Object getModelObject() {
    return element;
  }

  /*
   * Returns parent datapath view
   */
  public DatapathView getParentDatapathView() {
    return parentDatapathView;
  }

  /*
```

```
   * Returns tree node
   */
  public DefaultMutableTreeNode getTreeNode() {
    return treeNode;
  }

  /*
   * Sets graph cell
   */
  public void setGraphCell(DefaultGraphCell cell) {
    this.cell = cell;
  }

  /*
   * (non-Javadoc)
   *
   * @see java.lang.Object#toString()
   */
  public String toString() {
    return element.getName();
  }
}
```

B.7.2 DatapathView.java

```
package maaoha.view.model;

import java.awt.Color;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;
import java.awt.Point;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Map;
import java.util.Hashtable;
import java.util.Arrays;
import javax.swing.BorderFactory;
import javax.swing.JScrollPane;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.TreePath;
import javax.swing.tree.DefaultTreeModel;
import org.jgraph.JGraph;
import org.jgraph.graph.DefaultEdge;
import org.jgraph.graph.DefaultPort;
import org.jgraph.graph.DefaultGraphCell;
import org.jgraph.graph.DefaultGraphModel;
import org.jgraph.graph.GraphConstants;
import org.jgraph.graph.GraphLayoutCache;
import org.jgraph.graph.GraphModel;
import org.jgraph.graph.ParentMap;
import maaoha.MaaohaException;
import maaoha.model.Datapath;
import maaoha.model.DatapathElement;
import maaoha.model.Expression;
import maaoha.model.Module;
import maaoha.model.Sfg;
import maaoha.model.Model;
import maaoha.model.UseModuleAndParameter;
import maaoha.model.UseModule;
import maaoha.view.DatapathGraph;
import maaoha.view.TreeInfo;
import maaoha.view.MaaohaDatapathMarqueeHandler;
import maaoha.view.GPCellViewFactory;

/*
 * Class responsible for visualization of datapath elements
```

```
 */
public class DatapathView
{
  final static int topMargin = 50;
  final static int stepInputY = 50;
  final static int stepOutputY = 50;
  final static int stepElementY = 50;

  static int nextInputX = 0;
  static int nextOutputX = 700;
  static int nextElementX = 350;

  final static int sizeInputW = 80;
  final static int sizeInputH = 20;
  final static int sizeOutputW = 80;
  final static int sizeOutputH = 20;
  final static int sizeRegisterW = 40;
  final static int sizeRegisterH = 40;
  final static int sizeSignalW = 30;
  final static int sizeSignalH = 30;
  final static int sizeExpressionW = 160;
  final static int sizeExpressionH = 40;
  final static int sizeModuleW = 200;
  final static int sizeModuleH = 80;

  final static int routingOffset = 30;

  final static int ZombieExpressionX = 10;
  final static int ZombieExpressionY = 10 - topMargin;

  final static Color errorColor = new Color(255,0,0);
  final static Color inputColor = new Color(0,255,0);
  final static Color outputColor = new Color(128,0,255);
  final static Color registerColor = new Color(0,128,0);
  final static Color signalColor = new Color(192,192,192);
  final static Color expressionColor = new Color(255,255,0);
  final static Color moduleColor = new Color(128,128,255);

  protected DefaultMutableTreeNode inputsNode;

  protected DefaultMutableTreeNode outputsNode;

  protected DefaultMutableTreeNode registersNode;

  protected DefaultMutableTreeNode signalsNode;

  protected DefaultMutableTreeNode sfgsNode;

  protected DefaultMutableTreeNode treeNode;

  protected DefaultMutableTreeNode useModuleNode;

  protected Datapath datapath;

  protected ArrayList<DatapathElementView> elements;

  protected ArrayList<SfgView> sfgs;

  protected ArrayList<UseModuleView> useModules;

  protected JScrollPane view;

  protected JGraph dpGraph;

  protected double dpGraphScale;

  protected GraphModel dpModel;
```

```
protected GraphLayoutCache layoutCache;

protected ModuleView parent;

protected int nextInputY;

protected int nextOutputY;

protected int nextElementY;

protected Object currentView;

protected ArrayList<DefaultGraphCell> cellGroups;

/*
 * Constructor
 */
public DatapathView(Datapath datapath, ModuleView parent)
{
  this.datapath = datapath;
  this.parent = parent;
  currentView = null;

  elements = new ArrayList<DatapathElementView>();
  sfgs = new ArrayList<SfgView>();
  useModules = new ArrayList<UseModuleView>();
  cellGroups = new ArrayList<DefaultGraphCell>();

  for (int i = 0; i < datapath.getElementsNo(); i++)
  {
    elements.add(new DatapathElementView(datapath.getElement(i), this));
  }

  for (int i = 0; i < datapath.getUseModulesNo(); i++)
  {
    useModules.add(new UseModuleView(datapath.getUseModule(i), this));
  }

  for (int i = 0; i < datapath.getSfgsNo(); i++)
  {
    sfgs.add(new SfgView(datapath.getSfg(i), this));
  }

  dpGraphScale = 1;
}

/*
 * Adds arrows that are connected to the cell to the list
 */
protected void addArrowsToList(ArrayList<DefaultGraphCell> list,
  DefaultGraphCell cell)
{
  for (int i = 0; i < cell.getChildCount(); i++)
  {
    Object obj = cell.getChildAt(i);
    if (obj instanceof DefaultPort)
    {
      DefaultPort port = (DefaultPort) obj;
      Iterator iter = port.edges();
      while (iter.hasNext())
      {
        DefaultGraphCell gcell = (DefaultGraphCell) iter.next();
        if (!list.contains(gcell))
          list.add(gcell);
      }
    }
  }
}
```

```
/*
 * Adds new element view
 */
public void addNewDatapathElementView(DatapathElement de, Point pt)
  throws MaaohaException
{
  DatapathElementView dev = new DatapathElementView(de, this);
  elements.add(dev);
  if (de.getElementType().equals(Model.typeSignal))
  {
    ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
      insertNodeInto(dev.createTree(), signalsNode, signalsNode.getChildCount());

    DefaultGraphCell cell = null;
    if (pt == null)
      cell = createVertex(dev, dev.getElement().getElementType());
    else
      cell = createVertex(dev, pt.x, pt.y, dev.getElement().getElementType());
    dev.setGraphCell(cell);
    dpGraph.getGraphLayoutCache().insert(cell);
  }
  if (de.getElementType().equals(Model.typeRegister))
  {
    ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
      insertNodeInto(dev.createTree(), registersNode,
      registersNode.getChildCount());

    DefaultGraphCell cell = null;
    if (pt == null)
      cell = createVertex(dev, dev.getElement().getElementType());
    else
      cell = createVertex(dev, pt.x, pt.y, dev.getElement().getElementType());

    dev.setGraphCell(cell);
    dpGraph.getGraphLayoutCache().insert(cell);
  }
  if (de.getElementType().equals(Model.typeInput))
  {
    ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
      insertNodeInto(dev.createTree(), inputsNode, inputsNode.getChildCount());

    DefaultGraphCell cell = null;
    if (pt == null)
      cell = createVertex(dev, dev.getElement().getElementType());
    else
      cell = createVertex(dev, pt.x, pt.y, dev.getElement().getElementType());
    dev.setGraphCell(cell);
    dpGraph.getGraphLayoutCache().insert(cell);
  }
  if (de.getElementType().equals(Model.typeOutput))
  {
    ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
      insertNodeInto(dev.createTree(), outputsNode, outputsNode.getChildCount());

    DefaultGraphCell cell = null;
    if (pt == null)
      cell = createVertex(dev, dev.getElement().getElementType());
    else
      cell = createVertex(dev, pt.x, pt.y, dev.getElement().getElementType());
    dev.setGraphCell(cell);
    dpGraph.getGraphLayoutCache().insert(cell);
}
  TreePath path = new TreePath(dev.getTreeNode().getPath());
  parent.parent.parent.getTree().scrollPathToVisible(path);
  parent.parent.parent.getTree().setSelectionPath(path);

  checkModule();
```

```
  refreshView();
}

/*
 * Adds expression view
 */
public void addNewExpressionsView(ArrayList<Expression> expressionList,
  SfgView sfg ,Point pt) throws MaaohaException
{
  if (expressionList.size() > 0)
  {
    ArrayList<DefaultGraphCell> cells = new ArrayList<DefaultGraphCell>();

    for (int i = 0; i < expressionList.size(); i++)
    {
      ExpressionView ev = new ExpressionView(expressionList.get(i), sfg);
      sfg.expressions.add(ev);

      ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
        insertNodeInto(ev.createTree(), sfg.treeNode,
        sfg.treeNode.getChildCount());
      TreePath path = new TreePath(ev.getTreeNode().getPath());
      parent.parent.parent.getTree().scrollPathToVisible(path);
      parent.parent.parent.getTree().setSelectionPath(path);

      addSingleExpressionCells(ev, cells);
    }

    dpGraph.getGraphLayoutCache().insert(cells.toArray());
  }

  checkModule();
  setCurrentViewSfg(sfg);
}

/*
 * Creates and adds new sfg view
 */
public void addNewSfg(Sfg sfg) throws MaaohaException
{
  SfgView sfgView = new SfgView(sfg, this);
  sfgs.add(sfgView);
  sfgsNode.add(sfgView.createTree());
  parent.parent.parent.getTreeModel().reload();

  ArrayList<DefaultGraphCell> cells = new ArrayList<DefaultGraphCell>();
  addSingleSfgCells(sfgView, cells);

  dpGraph.getGraphLayoutCache().insert(cells.toArray());
  setCurrentViewSfg(sfgView);
}

/*
 * Adds vertex to the graph for single expression
 */
protected void addSingleExpressionCells(ExpressionView ev,
  ArrayList<DefaultGraphCell> cells) throws MaaohaException
{
  DefaultGraphCell cell = createVertex(ev, maaoha.model.Model.typeExpression);
  cells.add(cell);
  ev.setGraphCell(cell);

  DefaultEdge edge = new DefaultEdge();
  edge.setSource(ev.getGraphCell().getChildAt(1));
  DatapathElementView view =
    findElementView(ev.getExpression().getLeftSideElement());
  if (view != null)
  {
```

```
    edge.setTarget(view.getGraphCell().getChildAt(0));

    GraphConstants.setLineEnd(edge.getAttributes(),
      GraphConstants.ARROW_TECHNICAL);
    GraphConstants.setEndFill(edge.getAttributes(), true);
    GraphConstants.setLineStyle(edge.getAttributes(),
      GraphConstants.STYLE_ORTHOGONAL);
    GraphConstants.setBendable(edge.getAttributes(), true);
    cells.add(edge);
  }

  for (int k = 0; k < ev.getExpression().getRightSideElementsNo(); k++)
  {
    edge = new DefaultEdge();

    view = findElementView(ev.getExpression().getRightSideElement(k));
    if (view != null)
    {
      edge.setSource(view.getGraphCell().getChildAt(0));
      edge.setTarget(ev.getGraphCell().getChildAt(0));

      // add control points for default routing
      if ((ev.getExpression().getLeftSideElement().getElementType().
        equals(Model.typeRegister) ||zev.getExpression().getLeftSideElement().
        getElementType() == Model.typeSignal) &&
        (view.getElement().getElementType() == Model.typeRegister ||
        view.getElement().getElementType() == Model.typeSignal))
      {
        Map edgeAttributes  = new Hashtable();
        Map mapEv = ev.getGraphCell().getAttributes();
        Rectangle2D rect = GraphConstants.getBounds(mapEv);
        int trgy = (int) rect.getCenterY();
        int trgx = (int) rect.getMinX();
        mapEv = view.getGraphCell().getAttributes();
        rect = GraphConstants.getBounds(mapEv);
        int srcy = (int) rect.getCenterY();
        int srcx = (int) rect.getCenterX();

        GraphConstants.setPoints(edgeAttributes, Arrays.asList(new Object[] {
          view.getGraphCell().getChildAt(0), new Point( srcx,
          srcy - routingOffset + k * 4), new Point(trgx - routingOffset + k * 4,
          srcy - routingOffset + k * 4), new Point(trgx - routingOffset + k * 4,
          trgy), ev.getGraphCell().getChildAt(0) }));
        edge.getAttributes().applyMap(edgeAttributes);
        Hashtable<DefaultEdge, Map> nested = new Hashtable<DefaultEdge, Map>();
        nested.put(edge, edgeAttributes);
        layoutCache.insert(new Object[] { edge }, nested, null, null);
      }

      // set style
      GraphConstants.setLineEnd(edge.getAttributes(),
        GraphConstants.ARROW_TECHNICAL);
      GraphConstants.setEndFill(edge.getAttributes(), true);
      GraphConstants.setLineStyle(edge.getAttributes(),
        GraphConstants.STYLE_ORTHOGONAL);
      //GraphConstants.setRouting(edge.getAttributes(),
        GraphConstants.ROUTING_SIMPLE);
      GraphConstants.setBendable(edge.getAttributes(), true);

      cells.add(edge);
    }
  }
}

/*
 * Adds single sub-module cells to the graph
 */
protected void addSingleModuleCells(UseModuleView umv,
```

```
  ArrayList<DefaultGraphCell> cells) throws MaaohaException
{
  addSingleModuleCells(umv, cells, null);
}

/*
 * Adds single sub-module cells to the graph
 */
protected void addSingleModuleCells(UseModuleView umv,
  ArrayList<DefaultGraphCell> cells, Point pt) throws MaaohaException
{
  DefaultGraphCell cell;

  if (pt == null)
    cell = createVertex(umv, maaoha.model.Model.typeExpression);
  else
    cell = createVertex(umv, pt.x, pt.y, maaoha.model.Model.typeExpression);
  cells.add(cell);
  umv.setGraphCell(cell);
  Rectangle2D rectMod = GraphConstants.getBounds(cell.getAttributes());

  int lastInputY = 0;
  int lastOutputY = 0;

  for (int k = 0; k < umv.getUseModule().getParametersNo(); k++)
  {
    DefaultEdge edge = new DefaultEdge();
    DatapathElementView view =
      findElementView(umv.getUseModule().getParameter(k));
    if (view != null)
    {
      int width = 0;
      int height = 0;
      if (view.getElement().getElementType().equals(Model.typeSignal))
      {
        width = sizeSignalW;
        height = sizeSignalH;
      }
      if (view.getElement().getElementType().equals(Model.typeRegister))
      {
        width = sizeRegisterW;
        height = sizeRegisterH;
      }
      if (view.getElement().getElementType().equals(Model.typeInput))
      {
        width = sizeInputW;
        height = sizeInputH;
      }
      if (view.getElement().getElementType().equals(Model.typeOutput))
      {
        width = sizeOutputW;
        height = sizeOutputH;
      }
      if (umv.getUseModule().getModule().getDatapath().getInputOutputType(k).
        equals(Model.typeInput))
      {
        edge.setSource(view.getGraphCell().getChildAt(0));
        edge.setTarget(umv.cell.getChildAt(0));

        if (pt == null)
        {
          Rectangle2D rect = new Rectangle2D.Double(rectMod.getMinX() - 20 -
            width, rectMod.getMinY() + lastInputY ,width,height);
          GraphConstants.setBounds(view.getGraphCell().getAttributes(), rect);
          lastInputY += height + 20;
        }
      }
      else
```

```
      {
        edge.setSource(umv.cell.getChildAt(0));
        edge.setTarget(view.getGraphCell().getChildAt(0));

        if (pt == null)
        {
          Rectangle2D rect = new Rectangle2D.Double(rectMod.getMaxX() + 20,
            rectMod.getMinY() + lastOutputY,width,height);
          GraphConstants.setBounds(view.getGraphCell().getAttributes(), rect);
          lastOutputY += height + 20;
        }
      }

      GraphConstants.setLineEnd(edge.getAttributes(),
        GraphConstants.ARROW_TECHNICAL);
      GraphConstants.setEndFill(edge.getAttributes(), true);
      GraphConstants.setLineStyle(edge.getAttributes(),
        GraphConstants.STYLE_ORTHOGONAL);
      GraphConstants.setBendable(edge.getAttributes(), true);

      cells.add(edge);
    }
  }
}

/*
 * Adds single sfg cells to the graph
 */
protected void addSingleSfgCells(SfgView sfg, ArrayList<DefaultGraphCell> cells)
  throws MaaohaException
{
  for (int j = 0; j < sfg.getExpressionsNo(); j++)
    addSingleExpressionCells(sfg.expressions.get(j), cells);
}

/*
 * Creates and adds new sub-module view
 */
public void addUseModule(UseModule um, Point pt)  throws MaaohaException
{
  UseModuleView umv = new UseModuleView(um, this);
  useModules.add(umv);

  useModuleNode.add(umv.createTree());
  parent.parent.parent.getTreeModel().reload();
  ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
    insertNodeInto(umv.createTree(), useModuleNode,
    useModuleNode.getChildCount());
  TreePath path = new TreePath(umv.getTreeNode().getPath());
  parent.parent.parent.getTree().scrollPathToVisible(path);
  parent.parent.parent.getTree().setSelectionPath(path);

  ArrayList<DefaultGraphCell> cells = new ArrayList<DefaultGraphCell>();
  addSingleModuleCells(umv, cells, pt);

  dpGraph.getGraphLayoutCache().insert(cells.toArray());

  setCurrentViewDatapath();
}

/*
 * Checks module
 */
protected void checkModule()
{
  if (parent.getParent().getToolView().getVerifyModel())
  {
    try { parent.getParent().checkModel(); }
```

```
    catch (MaaohaException ex) {ex.toString(); }
  }
}

/*
 * Collapses group of cells
 */
public void colapseCellGroups()
{
  dpGraph.getGraphLayoutCache().collapse(cellGroups.toArray());
}

/*
 * Collapses graph cells
 */
public void colapseGraphCells()
{
  double gs2 = 2 * dpGraph.getGridSize();
  Object[] cells = dpGraph.getSelectionCells();
  Rectangle2D collapsedBounds = dpGraph.getCellBounds(cells);
  collapsedBounds.setFrame(collapsedBounds.getX(), collapsedBounds.getY(),
    Math.max(collapsedBounds.getWidth() / 4, gs2),
    Math.max(collapsedBounds.getHeight() / 2, gs2));
  dpGraph.snap(collapsedBounds);
  DefaultGraphCell group = createGroupCell(collapsedBounds);
  cellGroups.add(group);

  if (group != null && cells != null && cells.length > 0)
  {
    // Create the group structure
    ParentMap pm = new ParentMap();
    for (int i = 0; i < cells.length; i++)
    {
      pm.addEntry(cells[i], group);
      if (cells[i] instanceof DefaultGraphCell)
      {
        Object uobj = ((DefaultGraphCell) cells[i]).getUserObject();
        if (uobj instanceof DatapathElementView)
        {
          DatapathElementView dev = (DatapathElementView) uobj;
          for (int j = 0; j < sfgs.size(); j++)
          {
            for (int k = 0; k < sfgs.get(j).getExpressionsNo(); k++)
            {
              if (sfgs.get(j).getExpression(k).getExpression().
                getLeftSideElement().equals(dev.getElement()))
              {
                pm.addEntry(sfgs.get(j).getExpression(k).getGraphCell(), group);
              }
            }
          }
        }
      }
    }

    dpGraph.getGraphLayoutCache().insert(new Object[] { group }, null, null, pm);
    dpGraph.getGraphLayoutCache().collapse(new Object[] { group });
  }
}

/*
 * Groups cells for collapsing
 */
protected DefaultGraphCell createGroupCell(Rectangle2D collapsedBounds)
{
  DefaultGraphCell group = new DefaultGraphCell();
  group.addPort();
  GraphConstants.setInset(group.getAttributes(), 10);
```

```
  GraphConstants.setBackground(group.getAttributes(), new Color(240, 240, 255));
  GraphConstants.setBorderColor(group.getAttributes(), Color.black);
  GraphConstants.setOpaque(group.getAttributes(), true);
  GraphConstants.setBounds(group.getAttributes(), collapsedBounds);
  return group;
}

/*
 * Creates layout cache
 */
public GraphLayoutCache createLayoutCache(ArrayList<Sfg> sfgs)
{
  GraphLayoutCache layoutCache = new GraphLayoutCache(dpModel,
    new GPCellViewFactory(), true);

  return layoutCache;
}

/*
 * Creates sfg
 */
public void createSfg(String name, String expressionsToParse) throws
  MaaohaException
{
  Sfg sfg  = getDatapath().addSfgFromString(name, expressionsToParse);
  addNewSfg(sfg);
}

/*
 * Creates tree node
 */
public DefaultMutableTreeNode createTree()
{
  treeNode = new DefaultMutableTreeNode(new TreeInfo("Datapath", this));

  inputsNode = new DefaultMutableTreeNode("Inputs");
  outputsNode = new DefaultMutableTreeNode("Outputs");
  registersNode = new DefaultMutableTreeNode("Registers");
  signalsNode = new DefaultMutableTreeNode("Signals");


  for (int i = 0; i < elements.size(); i++)
  {
    String type = elements.get(i).getElement().getElementType();
    if (type.equals(maaoha.model.Model.typeInput))
    {
      inputsNode.add(elements.get(i).createTree());
    }
    else if (type.equals(maaoha.model.Model.typeOutput))
    {
      outputsNode.add(elements.get(i).createTree());
    }
    else if (type.equals(maaoha.model.Model.typeRegister))
    {
      registersNode.add(elements.get(i).createTree());
    }
    else if (type.equals(maaoha.model.Model.typeSignal))
    {
      signalsNode.add(elements.get(i).createTree());
    }
  }
  treeNode.add(inputsNode);
  treeNode.add(outputsNode);
  treeNode.add(registersNode);
  treeNode.add(signalsNode);

  useModuleNode = new DefaultMutableTreeNode("Use Module");
  for (int i = 0; i < useModules.size(); i++)
```

```
    {
      useModuleNode.add(useModules.get(i).createTree());
    }

    treeNode.add(useModuleNode);

    sfgsNode = new DefaultMutableTreeNode("Signal Flow Graphs");

    for (int i = 0; i < sfgs.size(); i++)
    {
      sfgsNode.add(sfgs.get(i).createTree());
    }

    treeNode.add(sfgsNode);

    return treeNode;
  }

  /*
   * Creates new sub-module
   */
  public void createUseModule(Point point, String name, Module selectedModule,
    String parameterList)  throws MaaohaException
  {
    UseModule um = datapath.addUseModule(name, selectedModule, parameterList);
    addUseModule(um, point);
  }

  /*
   * Creates vertex
   */
  protected DefaultGraphCell createVertex(Object obj, double x, double y,
    String cellType) throws MaaohaException
  {
    // Create vertex with the given name
    DefaultGraphCell cell = new DefaultGraphCell(obj);

    double w = 0;
    double h = 0;
    Color bg = null;
    boolean raised = true;

    if (cellType.equals(maaoha.model.Model.typeInput))
    {
      w = sizeInputW;//80;
      h = sizeInputH;//20;
      nextInputY += stepInputY + h;
      bg = inputColor;
      cell.addPort(new Point(GraphConstants.PERMILLE,
        GraphConstants.PERMILLE / 2));
    }
    else if (cellType.equals(maaoha.model.Model.typeOutput))
    {
      w = sizeOutputW;//80;
      h = sizeOutputH;//20;
      nextOutputY += stepOutputY + h;
      bg = outputColor;
      cell.addPort(new Point(0, GraphConstants.PERMILLE / 2));
    }
    else if (cellType.equals(maaoha.model.Model.typeRegister))
    {
      w = sizeRegisterW;//40;
      h = sizeRegisterH;//40;
      nextElementY += stepElementY + h;
      bg = registerColor;
      // Add a Floating Port
      cell.addPort();
    }
```

```
else if (cellType.equals(maaoha.model.Model.typeSignal))
{
  w = sizeSignalW;//30;
  h = sizeSignalH;//30;
  nextElementY += stepElementY + h;
  bg = signalColor;
  // Add a Floating Port
  cell.addPort();
}
else if (obj instanceof ExpressionView)
{
  ExpressionView exp = (ExpressionView) obj;
  DatapathElementView leftEl =
    findElementView(exp.expression.getLeftSideElement());
  if (leftEl != null)
  {
    Rectangle2D rect = GraphConstants.getBounds(leftEl.cell.getAttributes());
    x = rect.getMinX() - 200;
    y = rect.getMinY();
    y -=  topMargin;
  }
  else
  {
    x = ZombieExpressionX;
    y = ZombieExpressionY;
  }
  w = sizeExpressionW;
  h = sizeExpressionH;
  bg = expressionColor;
  GPCellViewFactory.setViewClass(cell.getAttributes(),
    "maaoha.view.JGraphEllipseView");
  cell.addPort(new Point(0, GraphConstants.PERMILLE / 2));
  cell.addPort(new Point(GraphConstants.PERMILLE,
    GraphConstants.PERMILLE / 2));

}
else if (obj instanceof UseModuleView)
{
  w = sizeModuleW;
  h = sizeModuleH;
  nextElementY += stepElementY + h;
  bg = moduleColor;
  // Add a Floating Port
  cell.addPort();
}

y +=  topMargin;

// Set bounds
GraphConstants.setBounds(cell.getAttributes(), new Rectangle2D.Double(
    x, y, w, h));

// Set fill color
if (bg != null)
{
  GraphConstants.setGradientColor(cell.getAttributes(), bg);
  GraphConstants.setOpaque(cell.getAttributes(), true);
}

// Set raised border
if (raised)
  GraphConstants.setBorder(cell.getAttributes(), BorderFactory
      .createRaisedBevelBorder());
else
  // Set black border
  GraphConstants.setBorderColor(cell.getAttributes(), Color.black);

for (int i = 0; i < useModules.size(); i++)
```

```
    {
      UseModule um = useModules.get(i).getUseModule();
      for (int j = 0; j < um.getParametersNo(); j++)
      {
        if (obj instanceof DatapathElementView)
          if (um.getParameter(j) == ((DatapathElementView) obj).getElement())
          {
            i = useModules.size();
            j = um.getParametersNo();
            nextElementY -= (stepElementY + h);
          }
      }
    }
    return cell;
  }

  /*
   * Creates vertex
   */
  protected DefaultGraphCell createVertex(Object obj, String cellType) throws
    MaaohaException
  {
    double x = 0;
    double y = 0;

    if (cellType.equals(maaoha.model.Model.typeInput))
    {
      x = nextInputX;
      y = nextInputY;
    }
    else if (cellType.equals(maaoha.model.Model.typeOutput))
    {
      x = nextOutputX;
      y = nextOutputY;
    }
    else if (cellType.equals(maaoha.model.Model.typeRegister))
    {
      x = nextElementX;
      y = nextElementY;
    }
    else if (cellType.equals(maaoha.model.Model.typeSignal))
    {
      x = nextElementX;
      y = nextElementY;
    }
    else if (obj instanceof ExpressionView)
    {
      ExpressionView exp = (ExpressionView) obj;
      DatapathElementView leftEl =
        findElementView(exp.expression.getLeftSideElement());
      if (leftEl != null)
      {
        Rectangle2D rect = GraphConstants.getBounds(leftEl.cell.getAttributes());
        x = rect.getMinX() - 200;
        y = rect.getMinY();
        y -=  topMargin;
      }
      else
      {
        x = 20;
        y = 20;
      }
    }
    else if (obj instanceof UseModuleView)
    {
      x = nextElementX;
      y = nextElementY;
    }
```

```
    y +=  topMargin;

    return createVertex(obj, x, y, cellType);
  }

  /*
   * Creates view
   */
  public JScrollPane createView() throws MaaohaException
  {
    dpModel = new DefaultGraphModel();
    layoutCache = new GraphLayoutCache(dpModel, new GPCellViewFactory(), true);
    dpGraph = new DatapathGraph(dpModel, layoutCache);

    // Control-drag should clone selection
    dpGraph.setCloneable(true);

    // Enable edit without final RETURN keystroke
    dpGraph.setInvokesStopCellEditing(true);

    // When over a cell, jump to its default port (we only have one, anyway)
    dpGraph.setJumpToDefaultPort(true);

    dpGraph.setAntiAliased(true);
    dpGraph.setGridVisible(true);

    dpGraph.setMarqueeHandler(new MaaohaDatapathMarqueeHandler(dpGraph, parent));

    ArrayList<DefaultGraphCell> cells = new ArrayList<DefaultGraphCell>();

    for (int i = 0; i < elements.size(); i++)
    {
      DefaultGraphCell cell = createVertex(elements.get(i),
        elements.get(i).getElement().getElementType());
      cells.add(cell);
      elements.get(i).setGraphCell(cell);
    }

    for (int i = 0; i < useModules.size(); i++)
    {
      UseModuleView umv = useModules.get(i);
      addSingleModuleCells(umv, cells);
    }

    for (int i = 0; i < sfgs.size(); i++)
    {
      SfgView sfg = sfgs.get(i);
      addSingleSfgCells(sfg, cells);
    }

    dpGraph.getGraphLayoutCache().insert(cells.toArray());
    dpGraph.getGraphLayoutCache().setVisible(cells.toArray(), false);

    ArrayList<DefaultGraphCell> dpeCells = new ArrayList<DefaultGraphCell>();

    for (int i = 0; i < cells.size(); i++)
    {
      if (cells.get(i).getUserObject() instanceof DatapathElementView)
        dpeCells.add(cells.get(i));
    }

    for (int i = 0; i < cells.size(); i++)
    {
      if (cells.get(i).getUserObject() instanceof UseModuleView)
        dpeCells.add(cells.get(i));
    }
```

```
  dpGraph.getGraphLayoutCache().setVisible(dpeCells.toArray(), true);
  view = new JScrollPane(dpGraph);

  return view;
}

/*
 * Edits sfg
 */
public void editSfg(String name, String expressions, SfgView sfgView) throws
  MaaohaException
{
  ArrayList<Expression> expressionList =
    datapath.createExpressionListFromString(expressions);

  ArrayList<DefaultGraphCell> cellsList = new ArrayList<DefaultGraphCell>();
  sfgView.setName(name);

  for (int i = 0; i < sfgView.getExpressionsNo(); i++)
  {

    ExpressionView expressionView = sfgView.getExpression(i);
    sfgView.removeExpression(expressionView);
    DefaultGraphCell cell = expressionView.getGraphCell();
    cellsList.add(cell);
    addArrowsToList(cellsList, cell);
  }
  dpGraph.getGraphLayoutCache().remove(cellsList.toArray());


  datapath.setSfgExpressions(sfgView.getSfg(), expressionList);

  sfgView.expressions = new ArrayList<ExpressionView>();
  for (int i = 0; i < sfgView.getSfg().getExpressionsNo(); i++)
  {
    sfgView.expressions.add(new ExpressionView(
      sfgView.getSfg().getExpression(i), sfgView));
  }

  cellsList = new ArrayList<DefaultGraphCell>();
  addSingleSfgCells(sfgView, cellsList);
  dpGraph.getGraphLayoutCache().insert(cellsList.toArray());

  sfgView.refreshTree();
  ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
    reload(sfgView.getTreeNode());
  TreePath path = new TreePath(sfgView.getTreeNode().getPath());
  parent.parent.parent.getTree().scrollPathToVisible(path);
  parent.parent.parent.getTree().setSelectionPath(path);

  refreshView();
}

/*
 * Edits sub-module
 */
public void editUseModule(String name, Module selectedModule,
                          String parameterList, UseModuleView umv)
{
  // edit model
  umv.getUseModule().setName(name);
  umv.getUseModule().setModule(selectedModule);
  umv.getUseModule().setParameters(
    datapath.parseUseModuleParameterList(parameterList));

  // edit view
  // remove old edges
  DefaultGraphCell cell = umv.getGraphCell();
```

```
  ArrayList<DefaultGraphCell> list = new ArrayList<DefaultGraphCell>();
  for (int i = 0; i < cell.getChildCount(); i++)
  {
    Object obj = cell.getChildAt(i);
    if (obj instanceof DefaultPort)
    {
      DefaultPort port = (DefaultPort) obj;
      Iterator iter = port.edges();
      while (iter.hasNext())
      {
        DefaultGraphCell gcell = (DefaultGraphCell) iter.next();
        if (!list.contains(gcell))
          list.add(gcell);
      }
    }
  }
  dpGraph.getGraphLayoutCache().remove(list.toArray());

  // add new edges
  ArrayList<DefaultGraphCell> cells = new ArrayList<DefaultGraphCell>();
  for (int k = 0; k < umv.getUseModule().getParametersNo(); k++)
  {
    //setLabelPosition(edge.getAttributes(), pt);
    DefaultEdge edge = new DefaultEdge();
    DatapathElementView view =
      findElementView(umv.getUseModule().getParameter(k));
    if (view != null)
    {
      if (umv.getUseModule().getModule().getDatapath().
          getInputOutputType(k).equals(Model.typeInput))
      {
        edge.setSource(view.getGraphCell().getChildAt(0));
        edge.setTarget(umv.cell.getChildAt(0));
      }
      else
      {
        edge.setSource(umv.cell.getChildAt(0));
        edge.setTarget(view.getGraphCell().getChildAt(0));
      }

      GraphConstants.setLineEnd(edge.getAttributes(),
        GraphConstants.ARROW_TECHNICAL);
      GraphConstants.setEndFill(edge.getAttributes(), true);
      GraphConstants.setLineStyle(edge.getAttributes(),
        GraphConstants.STYLE_ORTHOGONAL);
      //GraphConstants.setRouting(edge.getAttributes(),
        GraphConstants.ROUTING_SIMPLE);
      GraphConstants.setBendable(edge.getAttributes(), true);
      cells.add(edge);
    }
  }
  dpGraph.getGraphLayoutCache().insert(cells.toArray());
}

/*
 * Returns element view of the specified element
 */
protected DatapathElementView findElementView(DatapathElement de)
{
  for (int i = 0; i < elements.size(); i++)
  {
    if (elements.get(i).element == de)
      return elements.get(i);
  }
  return null;
}

/*
```

```
 * Returns view of the specified expression
 */
public ExpressionView findExpressionView(Expression expression)
{
  for (int i = 0; i < sfgs.size(); i++)
  {
    for (int j = 0; j < sfgs.get(i).getExpressionsNo(); j++)
    {
      if (sfgs.get(i).getExpression(j).getExpression() == expression)
        return sfgs.get(i).getExpression(j);
    }
  }
  return null;
}

/*
 * Returns view of the specified sfg
 */
protected SfgView findSfgView(Sfg sfg)
{
  for (int i = 0; i < sfgs.size(); i++)
  {
    if (sfgs.get(i).sfg == sfg)
      return sfgs.get(i);
  }
  return null;
}

/*
 * Returns view of the specified sub-module
 */
public UseModuleView findUseModuleView(UseModule um)
{
  for (int i = 0; i < useModules.size(); i++)
  {
    if (useModules.get(i).module == um)
      return useModules.get(i);
  }
  return null;
}

/*
 * Returns current view
 */
public Object getCurrentView()
{
  return currentView;
}

/*
 * Returns datapath
 */
public Datapath getDatapath()
{
  return datapath;
}

/*
 * Returns graph scale
 */
public double getDatapathGraphScale()
{
  return dpGraphScale;
}

/*
 * Returns edges between two cells
 */
```

```
public ArrayList<DefaultEdge> getEdgesBetweenCells(DefaultGraphCell cell1,
                                                   DefaultGraphCell cell2)
{
  ArrayList<DefaultEdge> list = new ArrayList<DefaultEdge>();
  for (int i = 0; i < cell1.getChildCount(); i++)
  {
    Object obj = cell1.getChildAt(i);
    if (obj instanceof DefaultPort)
    {
      DefaultPort port = (DefaultPort) obj;
      Iterator iter = port.edges();
      while (iter.hasNext())
      {
        DefaultEdge edge = (DefaultEdge) iter.next();

        Object source = dpModel.getParent(dpModel.getSource(edge));
        Object target = dpModel.getParent(dpModel.getTarget(edge));
        if ((source == cell1 && target == cell2) || (source == cell2 && target ==
          cell1))
          list.add(edge);
      }
    }
  }
  return list;
}

/*
 * Returns i-th element view
 */
public DatapathElementView getElementView(int i)
{
  return elements.get(i);
}

/*
 * Returns number of element views
 */
public int getElementViewNo()
{
  return elements.size();
}

/*
 * Returns expression cells
 */
public ArrayList<DefaultGraphCell> getExpressionsCells()
{
  ArrayList<DefaultGraphCell> cells = new ArrayList<DefaultGraphCell>();
  for (int i = 0; i < sfgs.size(); i++)
  {
    for (int j = 0; j < sfgs.get(i).expressions.size(); j++)
    {
      cells.add(sfgs.get(i).expressions.get(j).getGraphCell());
    }
  }
  return cells;
}

/*
 * Returns graph
 */
public JGraph getGraph()
{
  return dpGraph;
}
/*
 * Returns list of invalid cells
 */
```

```
public ArrayList<DefaultGraphCell> getInvalidCells(DatapathElement de,
                                                   ArrayList<Sfg> sfgsToCheck)
{
  ArrayList<DefaultGraphCell> cellsToMark = new ArrayList<DefaultGraphCell>();

  for (int i = 0; i < sfgs.size(); i++)
  {
    if (sfgsToCheck.contains(sfgs.get(i).getSfg()))
    {
      for (int j = 0; j < sfgs.get(i).getExpressionsNo(); j++)
      {
        ExpressionView expressionView = sfgs.get(i).getExpression(j);
        if (expressionView.getExpression().getLeftSideElement() == de)
        {
          if (!cellsToMark.contains(expressionView.getGraphCell()))
            cellsToMark.add(expressionView.getGraphCell());
        }
      }
    }
  }
  return cellsToMark;
}
/*
 * Returns parent module
 */
public ModuleView getParent()
{
  return parent;
}
/*
 * Returns i-th sfg
 */
public SfgView getSfg(int i)
{
  return sfgs.get(i);
}
/*
 * Returns index of sfg view
 */
public int getSfgIndex(SfgView sfgView)
{
  return sfgs.indexOf(sfgView);
}
/*
 * Returns expressions cells of specified sfgs
 */
public ArrayList<DefaultGraphCell> getSfgsExpressionsCells(ArrayList<Sfg>
                                                           usedSfgs)
{
  ArrayList<DefaultGraphCell> cells = new ArrayList<DefaultGraphCell>();
  for (int i = 0; i < sfgs.size(); i++)
  {
    if (usedSfgs.contains(sfgs.get(i).getSfg()))
      for (int j = 0; j < sfgs.get(i).expressions.size(); j++)
      {
        cells.add(sfgs.get(i).expressions.get(j).getGraphCell());
      }
  }
  return cells;
}
/*
 * Returns number of sfgs
 */
public int getSfgsNo()
{
  return sfgs.size();
}
/*
```

```
 * Returns i-th sub-module
 */
public UseModuleView getUseModuleView(int i)
{
  return useModules.get(i);
}
/*
 * Returns number of sub-module views
 */
public int getUseModuleViewNo()
{
  return useModules.size();
}
/*
 * Unmark all cells
 */
public void markAllCellsValid()
{
  Map nested = new Hashtable();
  for (int i = 0; i < elements.size(); i++)
  {
    if (elements.get(i).getElement().getElementType().equals(Model.typeOutput))
    {
      Map attrMap = new Hashtable();
      GraphConstants.setGradientColor(attrMap, outputColor);
      nested.put(elements.get(i).getGraphCell(), attrMap);
    }
  }

  for (int i = 0; i < sfgs.size(); i++)
    for (int j = 0; j < sfgs.get(i).getExpressionsNo(); j++)
    {
      Map attrMap = new Hashtable();
      GraphConstants.setGradientColor(attrMap, expressionColor);
      nested.put(sfgs.get(i).getExpression(j).getGraphCell(), attrMap);
    }

  for (int i = 0; i < useModules.size(); i++)
  {
    Map attrMap = new Hashtable();
    GraphConstants.setGradientColor(attrMap, moduleColor);
    nested.put(useModules.get(i).getGraphCell(), attrMap);
  }

  dpGraph.getGraphLayoutCache().edit(nested, null, null, null);
}
/*
 * Marks invalid cells
 */
public void markCellsInvalid(ArrayList<DefaultGraphCell> outputCells,
                             ArrayList<DefaultGraphCell> expressionCells,
                             ArrayList<DefaultGraphCell> useModuleCells)
{
  Map nested = new Hashtable();
  if (expressionCells != null)
    for (int i = 0; i < expressionCells.size(); i++)
    {
      Map attrMap = new Hashtable();
      GraphConstants.setGradientColor(attrMap, errorColor);
      nested.put(expressionCells.get(i), attrMap);
    }

  if (useModuleCells != null)
    for (int i = 0; i < useModuleCells.size(); i++)
    {
      Map attrMap = new Hashtable();
      GraphConstants.setGradientColor(attrMap, errorColor);
      nested.put(useModuleCells.get(i), attrMap);
```

```
    }

  if (outputCells != null)
    for (int i = 0; i < outputCells.size(); i++)
    {
      Map attrMap = new Hashtable();
      GraphConstants.setGradientColor(attrMap, errorColor);
      nested.put(outputCells.get(i), attrMap);
    }

  dpGraph.getGraphLayoutCache().edit(nested, null, null, null);
}
/*
 * Unmarks invalid cells
 */
public void markCellsValid(ArrayList<DefaultGraphCell> expressionCells,
                           ArrayList<DefaultGraphCell> useModuleCells)
{
  Map nested = new Hashtable();
  for (int i = 0; i < expressionCells.size(); i++)
  {
    Map attrMap = new Hashtable();
    GraphConstants.setGradientColor(attrMap, expressionColor);
    nested.put(expressionCells.get(i), attrMap);
  }

  for (int i = 0; i < useModuleCells.size(); i++)
  {
    Map attrMap = new Hashtable();
    GraphConstants.setGradientColor(attrMap, moduleColor);
    nested.put(useModuleCells.get(i), attrMap);
  }

  dpGraph.getGraphLayoutCache().edit(nested, null, null, null);
  dpGraph.getGraphLayoutCache().setVisible(expressionCells.toArray(), false);
}
/*
 * Refreshes view
 */
public void refreshView()
{
  if (currentView == null)
    setCurrentViewDatapath();
  else if (currentView instanceof FsmTransitionView)
  {
    FsmTransitionView transView = (FsmTransitionView) currentView;
    setCurrentViewTransition(transView);
  }
  else if (currentView instanceof SfgView)
  {
    SfgView sfgView = (SfgView) currentView;
    this.setCurrentViewSfg(sfgView);
  }
}
/*
 * Removes element view and returns list of attached edges
 */
public ArrayList<DefaultEdge> removeElement(DatapathElementView elementView)
{
  ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
    removeNodeFromParent(elementView.getTreeNode());
  ArrayList<UseModuleAndParameter> formerParameters =
    datapath.removeElement(elementView.getElement());
  ArrayList<DefaultEdge> edges = null;
  if (formerParameters != null)
  {
    for (int i = 0; i < formerParameters.size(); i++)
    {
```

```
        ModuleView modv =
          getParent().getParent().findModuleView(formerParameters.get(i).module);
        DatapathElementView dev =
          modv.getDatapath().findElementView(formerParameters.get(i).element);
        UseModuleView umv =
          modv.getDatapath().findUseModuleView(formerParameters.get(i).useModule);

        DefaultGraphCell devCell = dev.getGraphCell();
        DefaultGraphCell umvCell = umv.getGraphCell();

        edges = getEdgesBetweenCells(devCell, umvCell);
      }
    }
    elements.remove(elementView);

    return edges;
  }
  /*
   * Removes expression
   */
  public void removeExpression(ExpressionView expressionView)
  {
    for (int i = 0; i < sfgs.size(); i++)
    {
      for (int j = 0; j < sfgs.get(i).getExpressionsNo(); j++)
      {
        if (sfgs.get(i).getExpression(j) == expressionView)
        {
          sfgs.get(i).removeExpression(expressionView);
        }
      }
    }
  }
  /*
   * Removes sfg
   */
  public void removeSfg(SfgView sfgView)
  {
    // remove from model
    datapath.removeSfg(sfgView.getSfg());

    // remove graphical representation
    ArrayList<DefaultGraphCell> cellsList = new ArrayList<DefaultGraphCell>();
    for (int i = 0; i < sfgView.getExpressionsNo(); i++)
    {

      ExpressionView expressionView = sfgView.getExpression(i);
      sfgView.removeExpression(expressionView);
      DefaultGraphCell cell = expressionView.getGraphCell();
      cellsList.add(cell);
      addArrowsToList(cellsList, cell);
    }
    dpGraph.getGraphLayoutCache().remove(cellsList.toArray());

    // remove tree node
    ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
      removeNodeFromParent(sfgView.getTreeNode());
    sfgs.remove(sfgView);
  }
  /*
   * Removes sub-module
   */
  public void removeUseModule(UseModuleView useModuleV)
  {
    ((DefaultTreeModel) parent.parent.parent.getTree().getModel()).
      removeNodeFromParent(useModuleV.getTreeNode());
    useModules.remove(useModuleV);
    datapath.removeUseModule(useModuleV.getUseModule());
```

```
}
/*
 * Removes sub-module
 */
public void removeUseModulesForModule(ModuleView moduleView)
{
  ArrayList<DefaultGraphCell> cellsList = new ArrayList<DefaultGraphCell>();

  for (int i = 0; i < useModules.size(); i++)
  {
    if (useModules.get(i).getUseModule().getModule() == moduleView.getModule())
    {
      UseModuleView umv = useModules.get(i);
      cellsList.add(umv.getGraphCell());
      addArrowsToList(cellsList, umv.getGraphCell());
      removeUseModule(umv);
    }
  }
  dpGraph.getGraphLayoutCache().remove(cellsList.toArray());
}
/*
 * Sets current view as the datapath view
 */
public void setCurrentViewDatapath()
{
  currentView = null;

  markAllCellsValid();
  markCellsInvalid(null, null, parent.useModuleToMark);

  ArrayList<DefaultGraphCell> allCells = getExpressionsCells();
  dpGraph.getGraphLayoutCache().setVisible(allCells.toArray(), false);

  parent.parent.parent.setVisibleModule(getParent().getView());
}
/*
 * Sets current view as a sfg view
 */
public void setCurrentViewSfg(SfgView sfgView)
{
  currentView = sfgView;

  markAllCellsValid();

  ArrayList<DefaultGraphCell> moduleCells = new ArrayList<DefaultGraphCell>();
  moduleCells.addAll(parent.useModuleToMark);

  ArrayList<DefaultGraphCell> transModuleCells = sfgView.getInvalidUseModules();
  for (int i = 0; i < transModuleCells.size(); i++)
    if (!moduleCells.contains(transModuleCells.get(i)))
      moduleCells.add(transModuleCells.get(i));

  markCellsInvalid(null, sfgView.getInvalidExpressions(), moduleCells);

  ArrayList<DefaultGraphCell> cells = sfgView.getCells();
  ArrayList<DefaultGraphCell> allCells = getExpressionsCells();

  dpGraph.getGraphLayoutCache().setVisible(allCells.toArray(), false);
  dpGraph.getGraphLayoutCache().setVisible(cells.toArray(), true);
  parent.parent.parent.setVisibleModule(getParent().getView());
}
/*
 * Sets current view as a transition view
 */
public void setCurrentViewTransition(FsmTransitionView transView)
{
  currentView = transView;
```

```
    markAllCellsValid();
    ArrayList<DefaultGraphCell> moduleCells = new ArrayList<DefaultGraphCell>();
    moduleCells.addAll(parent.useModuleToMark);

    ArrayList<DefaultGraphCell> transModuleCells =
      transView.getInvalidUseModules();
    for (int i = 0; i < transModuleCells.size(); i++)
      if (!moduleCells.contains(transModuleCells.get(i)))
        moduleCells.add(transModuleCells.get(i));

    markCellsInvalid(transView.getInvalidOutputs(),
      transView.getInvalidExpressions(), moduleCells);

    ArrayList<DefaultGraphCell> allCells = getExpressionsCells();
    ArrayList<DefaultGraphCell> cells =
      getSfgsExpressionsCells(transView.getTransition().getSfgs());

    dpGraph.getGraphLayoutCache().setVisible(allCells.toArray(), false);
    dpGraph.getGraphLayoutCache().setVisible(cells.toArray(), true);

    parent.parent.parent.setVisibleModule(getParent().getView());
  }
  /*
   * Sets graph scale
   */
  public void setDatapathGraphScale(double scale, Point2D.Double pt)
  {
    if (pt == null)
      dpGraph.setScale(scale);
    else
      dpGraph.setScale(scale, pt);
    dpGraphScale = scale;
  }
}
```

B.7.3 AddDatapathElementDialog.java

```
package maaoha.view.model;

import javax.swing.tree.DefaultMutableTreeNode;

import maaoha.model.Expression;
import maaoha.view.TreeInfo;

import org.jgraph.graph.DefaultGraphCell;

/*
 * Class responsible for visualization of expression
 */
public class ExpressionView implements GraphCellView {
  protected DefaultMutableTreeNode treeNode;

  protected DefaultGraphCell cell;

  protected Expression expression;

  protected SfgView parent;

  /*
   * Constructor
   */
  public ExpressionView(Expression expression, SfgView parent) {
    this.expression = expression;
    this.parent = parent;
  }
```

```
  /*
   * Creates tree node
   */
  public DefaultMutableTreeNode createTree() {
    treeNode = new DefaultMutableTreeNode(new TreeInfo(expression
        .getValue(), this));
    return treeNode;
  }

  /*
   * Returns expression
   */
  public Expression getExpression() {
    return expression;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.view.model.GraphCellView#getGraphCell()
   */
  public DefaultGraphCell getGraphCell() {
    return cell;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.view.model.GraphCellView#getModelObject()
   */
  public Object getModelObject() {
    return expression;
  }

  /*
   * Returns parent sfg view
   */
  public SfgView getParent() {
    return parent;
  }

  /*
   * Returns tree node
   */
  public DefaultMutableTreeNode getTreeNode() {
    return treeNode;
  }

  /*
   * Sets graph cell
   */
  public void setGraphCell(DefaultGraphCell cell) {
    this.cell = cell;
  }

  /*
   * (non-Javadoc)
   *
   * @see java.lang.Object#toString()
   */
  public String toString() {
    return expression.getValue();
  }
}
```

B.7.4 FsmStateView.java

```
package maaoha.view.model;

import java.util.ArrayList;

import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;

import maaoha.model.FsmState;
import maaoha.view.TreeInfo;

import org.jgraph.graph.DefaultGraphCell;

/*
 * Class responsible for visualization of controller state
 */
public class FsmStateView implements GraphCellView {
  protected ArrayList<FsmTransitionView> transitions;

  protected FsmState state;

  protected FsmView parent;

  protected DefaultGraphCell cell;

  protected DefaultMutableTreeNode treeNode;

  /*
   * Constructor
   */
  public FsmStateView(FsmState state, FsmView parent) {
    this.state = state;
    this.parent = parent;

    transitions = new ArrayList<FsmTransitionView>();

    for (int i = 0; i < state.getTransitionsNo(); i++) {
      transitions
          .add(new FsmTransitionView(state.getTransistion(i), this));
    }
  }

  /*
   * Adds transition
   */
  public void addFsmTransition(FsmTransitionView transView) {
    transitions.add(transView);
  }

  /*
   * Adds transition, it's tree node and sets its parent
   */
  public void addFsmTransitionComplete(FsmTransitionView transView) {
    transitions.add(transView);
    transView.setParent(this);
    ((DefaultTreeModel) parent.getParent().getParent().getToolView()
        .getTree().getModel()).insertNodeInto(transView.createTree(),
        treeNode, treeNode.getChildCount());
  }

  /*
   * Creates tree node
   */
  public DefaultMutableTreeNode createTree() {
    treeNode = new DefaultMutableTreeNode(new TreeInfo(state.getName(),
        this));
```

```
    for (int i = 0; i < transitions.size(); i++) {
      treeNode.add(transitions.get(i).createTree());
    }

    return treeNode;
  }

  /*
   * Returns controller state
   */
  public FsmState getFsmState() {
    return state;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.view.model.GraphCellView#getGraphCell()
   */
  public DefaultGraphCell getGraphCell() {
    return cell;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.view.model.GraphCellView#getModelObject()
   */
  public Object getModelObject() {
    return state;
  }

  /*
   * Returns parent controller view
   */
  public FsmView getParent() {
    return parent;
  }

  /*
   * Returns i-th transition view
   */
  public FsmTransitionView getTransitionView(int i) {
    return transitions.get(i);
  }

  /*
   * Returns index of transition view
   */
  public int getTransitionViewIndex(FsmTransitionView transView) {
    return transitions.indexOf(transView);
  }

  /*
   * Returns number of tranisition views
   */
  public int getTransitionViewNo() {
    return transitions.size();
  }

  /*
   * Returns tree node
   */
  public DefaultMutableTreeNode getTreeNode() {
    return treeNode;
  }

  /*
```

```
   * Removes state from parent
   */
  public void remove() {
    parent.removeFsmState(this, treeNode);
  }

  /*
   * Removes transition
   */
  public void removeTransition(FsmTransitionView transView) {
    ((DefaultTreeModel) parent.parent.parent.parent.getTree().getModel())
        .removeNodeFromParent(transView.getTreeNode());
    transitions.remove(transView);
    state.removeTransition(transView.getTransition());
  }

  /*
   * Sets graph cell
   */
  public void setGraphCell(DefaultGraphCell cell) {
    this.cell = cell;
  }

  /*
   * (non-Javadoc)
   *
   * @see java.lang.Object#toString()
   */
  public String toString() {
    return state.getName();
  }
}
```

B.7.5 FsmTransitionView.java

```
package maaoha.view.model;

import java.util.ArrayList;

import javax.swing.tree.DefaultMutableTreeNode;

import maaoha.model.FsmTransition;
import maaoha.view.ToolView;
import maaoha.view.TreeInfo;

import org.jgraph.graph.DefaultGraphCell;
import org.jgraph.graph.GraphLayoutCache;

/*
 * Class responsible for visualization of controller transition
 */
public class FsmTransitionView {
  protected ArrayList<DefaultGraphCell> invalidUseModules;

  protected ArrayList<DefaultGraphCell> invalidOutputs;

  protected ArrayList<DefaultGraphCell> invalidExpressions;

  protected DefaultGraphCell cell;

  protected FsmTransition transition;

  protected FsmStateView parent;

  protected DefaultMutableTreeNode treeNode;
```

```
protected GraphLayoutCache layoutCache;

/*
 * Constructor
 */
public FsmTransitionView(FsmTransition transition, FsmStateView parent) {
  this.parent = parent;
  this.transition = transition;

  invalidOutputs = new ArrayList<DefaultGraphCell>();
  invalidExpressions = new ArrayList<DefaultGraphCell>();
  invalidUseModules = new ArrayList<DefaultGraphCell>();
}

/*
 * Creates tree node
 */
public DefaultMutableTreeNode createTree() {
  treeNode = new DefaultMutableTreeNode(new TreeInfo(
      transition.getName(), this));
  return treeNode;
}

/*
 * Returns graph cell
 */
public DefaultGraphCell getGraphCell() {
  return cell;
}

/*
 * Returns invalid expressions
 */
public ArrayList<DefaultGraphCell> getInvalidExpressions() {
  return invalidExpressions;
}

/*
 * Returns invalid outputs
 */
public ArrayList<DefaultGraphCell> getInvalidOutputs() {
  return invalidOutputs;
}

/*
 * Returns invalid sub-modules
 */
public ArrayList<DefaultGraphCell> getInvalidUseModules() {
  return invalidUseModules;
}

/*
 * Returns number of prior transitions to the same destination
 */
public int getNumberOfPriorTransitionToDestination() {
  return parent.getFsmState().getNumberOfTransitionToDestination(
      transition.getNextState(),
      parent.getFsmState().getTransitionIndex(transition));
}

/*
 * Returns parent controller state
 */
public FsmStateView getParent() {
  return parent;
}

/*
```

```
 * Returns transition
 */
public FsmTransition getTransition() {
  return transition;
}

/*
 * Returns tree node
 */
public DefaultMutableTreeNode getTreeNode() {
  return treeNode;
}

/*
 * Sets graph cell
 */
public void setGraphCell(DefaultGraphCell cell) {
  this.cell = cell;
}

/*
 * Sets transition invalid expression
 */
public void setInvalidExpressions(
    ArrayList<DefaultGraphCell> invalidExpressions) {
  this.invalidExpressions = invalidExpressions;
}

/*
 * Sets transition invalid outputs
 */
public void setInvalidOutputs(ArrayList<DefaultGraphCell> invalidOutputs) {
  this.invalidOutputs = invalidOutputs;
}

/*
 * Sets transition invalid sub-module
 */
public void setInvalidUseModules(
    ArrayList<DefaultGraphCell> invalidUseModules) {
  this.invalidUseModules = invalidUseModules;
}

/*
 * Sets parent controller state
 */
public void setParent(FsmStateView parent) {
  this.parent = parent;
}

/*
 * (non-Javadoc)
 *
 * @see java.lang.Object#toString()
 */
public String toString() {
  ToolView app = parent.parent.parent.parent.getToolView();
  String label = "";
  if (app.getDisplayIndexOnEdge()) {
    label += (parent.getFsmState().getTransitionIndex(transition) + 1)
        + ". ";
  }

  if (app.getDisplayConditionOnEdge()) {
    if (app.getDisplaySfgsOnEdge())
      label += "if ";
    label += transition.getCondition().getBasicExpression();
  }
```

```
    if (app.getDisplaySfgsOnEdge()) {
      if (app.getDisplayConditionOnEdge())
        label += " then ";
      label += "(" + transition.getSfgsString() + ")";
    }
    return label;
  }
}
```

B.7.6 FsmView.java

```
package maaoha.view.model;

import java.awt.Color;
import java.awt.Point;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Map;

import javax.swing.BorderFactory;
import javax.swing.JScrollPane;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.TreePath;

import maaoha.MaaohaException;
import maaoha.model.Fsm;
import maaoha.model.FsmState;
import maaoha.model.FsmTransition;
import maaoha.model.Sfg;
import maaoha.view.ControllerGraphSelectionListener;
import maaoha.view.FsmGraph;
import maaoha.view.GPCellViewFactory;
import maaoha.view.MaaohaControllerMarqueeHandler;
import maaoha.view.TreeInfo;

import org.jgraph.JGraph;
import org.jgraph.graph.ConnectionSet;
import org.jgraph.graph.DefaultEdge;
import org.jgraph.graph.DefaultGraphCell;
import org.jgraph.graph.DefaultGraphModel;
import org.jgraph.graph.DefaultPort;
import org.jgraph.graph.GraphConstants;
import org.jgraph.graph.GraphLayoutCache;
import org.jgraph.graph.GraphModel;

/*
 * Class responsible for visualization of controller
 */
public class FsmView {
  final static int topMargin = 50;
  final static int leftMargin = 100;
  final static int stepX = 500;
  final static int stepY = 200;
  final static int cellsInRow = 2;
  final static int distanceBetweenText = 120;
  final static Color errorColor = new Color(255, 0, 0);
  final static Color transitionColor = new Color(0, 0, 0);

  protected DefaultMutableTreeNode treeNode;

  protected Fsm fsm;

  protected ArrayList<FsmStateView> states;
```

```
protected JScrollPane view;

protected JGraph fsmGraph;

protected double fsmGraphScale;

protected GraphModel fsmModel;

protected GraphLayoutCache layoutCache;

protected ModuleView parent;

protected int nextElementIndex;

/*
 * Constructor
 */
public FsmView(Fsm fsm, ModuleView parent) {
  this.fsm = fsm;
  this.parent = parent;

  states = new ArrayList<FsmStateView>();

  for (int i = 0; i < fsm.getStatesNo(); i++) {
    states.add(new FsmStateView(fsm.getFsmState(i), this));
  }

  nextElementIndex = 0;
  fsmGraphScale = 1;
}

/*
 * Creates mew view for state and adds it
 */
public void addNewState(FsmState state, Point point) {
  FsmStateView stateV = new FsmStateView(state, this);

  // add graph cell
  DefaultGraphCell cell = new DefaultGraphCell();
  if (point != null)
    cell = createVertex(stateV, point.x, point.y);
  else
    cell = createVertex(stateV);
  fsmGraph.getGraphLayoutCache().insert(cell);

  stateV.setGraphCell(cell);

  // add tree entry
  ((DefaultTreeModel) parent.parent.parent.getTree().getModel())
      .insertNodeInto(stateV.createTree(), treeNode, treeNode
          .getChildCount());
  TreePath path = new TreePath(stateV.getTreeNode().getPath());
  parent.parent.parent.getTree().scrollPathToVisible(path);
  parent.parent.parent.getTree().setSelectionPath(path);

  states.add(stateV);
}

/*
 * Creates view for transition and adds it
 */
public void addNewTransition(FsmStateView stateV, FsmTransition trans)
    throws MaaohaException {
  FsmTransitionView transView = new FsmTransitionView(trans, stateV);
  stateV.addFsmTransition(transView);

  DefaultEdge edge = new DefaultEdge(transView);
```

```
  edge.setSource(stateV.cell.getChildAt(0));
  FsmStateView view = findView(transView.transition.getNextState());
  if (view == null)
    throw new MaaohaException("Can't find view");
  edge.setTarget(view.getGraphCell().getChildAt(0));

  Point2D pt = new Point2D.Double();
  pt.setLocation(GraphConstants.PERMILLE * 3 / 8 - distanceBetweenText
      * transView.getNumberOfPriorTransitionToDestination(), 0);

  int arrow = GraphConstants.ARROW_CLASSIC;
  GraphConstants.setLineEnd(edge.getAttributes(), arrow);
  GraphConstants.setEndFill(edge.getAttributes(), true);
  GraphConstants.setLabelPosition(edge.getAttributes(), pt);

  fsmGraph.getGraphLayoutCache().insert(edge);

  transView.setGraphCell(edge);

  ((DefaultTreeModel) parent.parent.parent.getTree().getModel())
      .insertNodeInto(transView.createTree(), stateV.getTreeNode(),
          stateV.getTreeNode().getChildCount());
  TreePath path = new TreePath(transView.getTreeNode().getPath());
  parent.parent.parent.getTree().scrollPathToVisible(path);
  parent.parent.parent.getTree().setSelectionPath(path);
}

/*
 * Creates tree node
 */
public DefaultMutableTreeNode createTree() {
  treeNode = new DefaultMutableTreeNode(new TreeInfo("Fsm", this));

  for (int i = 0; i < states.size(); i++) {
    treeNode.add(states.get(i).createTree());
  }

  return treeNode;
}

/*
 * Creates vertex
 */
protected DefaultGraphCell createVertex(Object obj) {
  double x = leftMargin + (nextElementIndex % cellsInRow) * stepX;
  double y = topMargin + (nextElementIndex / cellsInRow) * stepY;
  nextElementIndex++;

  return createVertex(obj, x, y);
}

/*
 * Creates vertex
 */
protected DefaultGraphCell createVertex(Object obj, double x, double y) {
  double w = 50;
  double h = 50;
  Color bg = new Color(141, 71, 141);
  boolean raised = true;

  // Create vertex with the given name
  DefaultGraphCell cell = new DefaultGraphCell(obj);

  GPCellViewFactory.setViewClass(cell.getAttributes(),
      "maaoha.view.JGraphEllipseView");

  // Set bounds
  GraphConstants.setBounds(cell.getAttributes(), new Rectangle2D.Double(
```

```
        x, y, w, h));

    // Set fill color
    if (bg != null) {
      GraphConstants.setGradientColor(cell.getAttributes(), bg);
      GraphConstants.setOpaque(cell.getAttributes(), true);
    }

    // Set raised border
    if (raised)
      GraphConstants.setBorder(cell.getAttributes(), BorderFactory
          .createRaisedBevelBorder());
    else
      // Set black border
      GraphConstants.setBorderColor(cell.getAttributes(), Color.black);

    // Add a Floating Port
    cell.addPort();

    return cell;
  }

  /*
   * Creates view
   */
  public JScrollPane createView() throws MaaohaException {
    fsmModel = new DefaultGraphModel();
    layoutCache = new GraphLayoutCache(fsmModel, new GPCellViewFactory(),
        false);
    fsmGraph = new FsmGraph(fsmModel, layoutCache);
    ControllerGraphSelectionListener listener = new
      ControllerGraphSelectionListener(
        parent);

    fsmGraph.addGraphSelectionListener(listener);

    // Control-drag should clone selection
    fsmGraph.setCloneable(true);

    // Enable edit without final RETURN keystroke
    fsmGraph.setInvokesStopCellEditing(true);

    // When over a cell, jump to its default port (we only have one, anyway)
    fsmGraph.setJumpToDefaultPort(true);

    fsmGraph.setAntiAliased(true);
    fsmGraph.setGridVisible(true);

    fsmGraph.setMarqueeHandler(new MaaohaControllerMarqueeHandler(fsmGraph,
        parent));

    int cellsNo = states.size() + fsm.getTotalTransitionsNo();// elements.size()
                                  // + 2 *
                                  // datapath.GetTotalExpressionNo()
                                  // +
                                  // datapath.GetTotalRightSideElementsNo();//
                                  // +
                                  // useModules.size();

    if (cellsNo > 0) {
      DefaultGraphCell[] cells = new DefaultGraphCell[cellsNo];
      int cellIndex = 0;

      for (int i = 0; i < states.size(); i++) {
        cells[cellIndex] = createVertex(states.get(i));
        states.get(i).setGraphCell(cells[cellIndex]);
        cellIndex++;
      }
```

```
    for (int i = 0; i < states.size(); i++) {
      FsmStateView state = states.get(i);
      for (int j = 0; j < state.transitions.size(); j++) {
        FsmTransitionView transV = state.transitions.get(j);
        DefaultEdge edge = new DefaultEdge(transV);
        transV.setGraphCell(edge);
        edge.setSource(state.cell.getChildAt(0));
        FsmStateView view = findView(state.transitions.get(j).transition
            .getNextState());
        if (view == null)
          throw new MaaohaException(
              "FsmView->CreateView::Can't find view");
        edge.setTarget(view.getGraphCell().getChildAt(0));

        Point2D pt = new Point2D.Double();
        pt.setLocation(GraphConstants.PERMILLE
            * 3
            / 8
            - distanceBetweenText
            * state.transitions.get(j)
                .getNumberOfPriorTransitionToDestination(),
            0);

        int arrow = GraphConstants.ARROW_CLASSIC;
        GraphConstants.setLineEnd(edge.getAttributes(), arrow);
        GraphConstants.setEndFill(edge.getAttributes(), true);
        GraphConstants.setLabelPosition(edge.getAttributes(), pt);

        cells[cellIndex++] = edge;
      }
    }
    fsmGraph.getGraphLayoutCache().insert(cells);
  }
  view = new JScrollPane(fsmGraph);

  return view;
}

/*
 * Updates transition
 */
public void editTransition(FsmStateView selectedSource,
    FsmStateView selectedTarget, String conditionsText,
    String sfgsText, FsmTransitionView transView)
    throws MaaohaException {
  Map edgeAttributes = new Hashtable();
  DefaultEdge edge = (DefaultEdge) transView.getGraphCell();
  Map m = edge.getAttributes();

  DefaultPort sourcePort = (DefaultPort) edge.getSource();
  DefaultPort targetPort = (DefaultPort) edge.getTarget();
  ;

  if (selectedSource != transView.getParent()) {
    transView.getParent().removeTransition(transView);
    // transView.getParent().getFsmState().
       removeTransition(transView.getTransition());
    selectedSource.addFsmTransitionComplete(transView);
    selectedSource.getFsmState().addTransition(
        transView.getTransition());
    transView.getGraphCell();
    sourcePort = (DefaultPort) selectedSource.getGraphCell()
        .getChildAt(0);
  }

  transView.getTransition().getCondition().setConditionFromString(
      conditionsText,
```

```
        parent.getDatapath().getDatapath().getElements(),
        parent.getDatapath().getDatapath().getLookupTables());

    ArrayList<Sfg> sfgs = parent.getModule().getSfgsFromString(sfgsText);
    parent.getModule().removeRedudantFromList(sfgs);
    transView.getTransition().setSfgs(sfgs);

    if (transView.getTransition().getNextState() != selectedTarget
        .getFsmState()) {
      // change target state
      transView.getTransition()
          .setNextState(selectedTarget.getFsmState());
      targetPort = (DefaultPort) selectedTarget.getGraphCell()
          .getChildAt(0);
    }

    ConnectionSet cSet = new ConnectionSet(edge, sourcePort, targetPort);

    fsmGraph.getGraphLayoutCache().edit(null, cSet, null, null);
  }

  /*
   * Returns view of controller state
   */
  protected FsmStateView findView(FsmState state) {
    for (int i = 0; i < states.size(); i++) {
      if (states.get(i).state == state) {
        return states.get(i);
      }
    }
    return null;
  }

  /*
   * Returns graph scale
   */
  public double getFsmGraphScale() {
    return fsmGraphScale;
  }

  /*
   * Returns i-th state view
   */
  public FsmStateView getFsmState(int i) {
    return states.get(i);
  }

  /*
   * Returns number of controller state views
   */
  public int getFsmStateNo() {
    return states.size();
  }

  /*
   * Returns view for controller state
   */
  public FsmStateView getFsmStateView(FsmState state) {
    for (int i = 0; i < states.size(); i++) {
      if (state == states.get(i).getFsmState())
        return states.get(i);
    }
    return null;
  }

  /*
   * Returns transition by global index
   */
```

```
public FsmTransitionView getFsmTransitionByTotalIndex(int index) {
  for (int i = 0; i < states.size(); i++) {
    if (index > states.get(i).getTransitionViewNo())
      index -= states.get(i).getTransitionViewNo();
    else {
      for (int j = 0; j < states.get(i).getTransitionViewNo(); j++) {
        if (index == 0)
          return states.get(i).getTransitionView(j);
        else
          index--;
      }
    }
  }
  return null;
}

/*
 * Returns graph
 */
public JGraph getGraph() {
  return fsmGraph;
}

/*
 * Returns parent module view
 */
public ModuleView getParent() {
  return parent;
}

/*
 * Returns transition view index
 */
public int getTransitionViewIndex(FsmTransitionView transView) {
  int totalIndex = 0;
  for (int i = 0; i < states.size(); i++) {
    int localIndex = states.get(i).getTransitionViewIndex(transView);
    if (localIndex > -1)
      return totalIndex + localIndex;

    totalIndex += states.get(i).getTransitionViewNo();
  }
  return -1;
}

/*
 * Marks cells invalid
 */
public void markCellsInvalid(ArrayList<DefaultGraphCell> cells) {
  Map nested = new Hashtable();
  for (int i = 0; i < cells.size(); i++) {
    Map attrMap = new Hashtable();
    GraphConstants.setForeground(attrMap, errorColor);
    nested.put(cells.get(i), attrMap);
  }

  fsmGraph.getGraphLayoutCache().edit(nested, null, null, null);
}

/*
 * Marks cells valid
 */
public void markCellsValid(ArrayList<DefaultGraphCell> cells) {
  Map nested = new Hashtable();
  for (int i = 0; i < cells.size(); i++) {
    Map attrMap = new Hashtable();
    GraphConstants.setForeground(attrMap, transitionColor);
    nested.put(cells.get(i), attrMap);
```

```
    }

    fsmGraph.getGraphLayoutCache().edit(nested, null, null, null);
  }

  /*
   * Removes controller state
   */
  public void removeFsmState(FsmStateView fsmStateView,
      DefaultMutableTreeNode node) {
    states.remove(fsmStateView);
    fsm.removeFsmState(fsmStateView.getFsmState());
    ((DefaultTreeModel) parent.parent.parent.getTree().getModel())
        .removeNodeFromParent(node);
  }

  /*
   * Controller part of removeSfg from ModuleView removes completely sfgView
   * from ModelView
   */
  protected void removeSfg(SfgView sfgView) {
    for (int i = 0; i < states.size(); i++) {
      states.get(i).getFsmState().removeSfg(sfgView.getSfg());
    }
  }

  /*
   * Sets graph scale
   */
  public void setFsmGraphScale(double scale, Point2D.Double pt) {
    if (pt == null)
      fsmGraph.setScale(scale);
    else
      fsmGraph.setScale(scale, pt);
    fsmGraphScale = scale;
  }
}
```

B.7.7 GraphCellView.java

```
package maaoha.view.model;

import org.jgraph.graph.DefaultGraphCell;

/*
 * Interface for all class responsible for visualization
 * of model elements that are represented in the graph by cell
 */
public interface GraphCellView {
  /*
   * Returns graph cell
   */
  public DefaultGraphCell getGraphCell();

  /*
   * Returns object
   */
  public Object getModelObject();
}
```

B.7.8 ModelView.java

```
package maaoha.view.model;

import java.awt.Component;
```

```
import java.awt.geom.Point2D;
import java.util.ArrayList;

import javax.swing.JDialog;
import javax.swing.JSplitPane;
import javax.swing.JTabbedPane;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;

import maaoha.MaaohaException;
import maaoha.model.Model;
import maaoha.model.Module;
import maaoha.view.AddDatapathElementDialog;
import maaoha.view.AddExpressionDialog;
import maaoha.view.AddFsmStateDialog;
import maaoha.view.AddFsmTransitionDialog;
import maaoha.view.ToolView;
import maaoha.view.UseModuleDialog;

/*
 * Class responsible for visualization of model
 */
public class ModelView {
  protected DefaultMutableTreeNode treeNode;

  protected ArrayList<ModuleView> moduleViews;

  protected Model model;

  protected ToolView parent;

  /*
   * Constructor
   */
  public ModelView(Model model, ToolView parent) {
    this.parent = parent;
    moduleViews = new ArrayList<ModuleView>();
    this.model = model;

    for (int i = 0; i < model.getModulesNo(); i++) {
      ModuleView moduleView = new ModuleView(model.getModule(i), this);
      moduleViews.add(moduleView);
    }
  }

  /*
   * Adds new graph element
   */
  public void addNewGraphElement(Component comp, String type)
      throws MaaohaException {
    for (int i = 0; i < moduleViews.size(); i++)
      if (moduleViews.get(i).getView() == comp) {
        JDialog cd = null;
        if (type.equals(Model.typeSignal)
            || type.equals(Model.typeRegister)
            || type.equals(Model.typeInput)
            || type.equals(Model.typeOutput))
          cd = new AddDatapathElementDialog(maaoha.MyTool.frame,
              type, moduleViews.get(i), null);
        else if (type.equals(Model.typeExpression))
          cd = new AddExpressionDialog(maaoha.MyTool.frame,
              moduleViews.get(i), null);
        else if (type.equals(Model.typeState))
          cd = new AddFsmStateDialog(maaoha.MyTool.frame, moduleViews
              .get(i), null);
        else if (type.equals(Model.typeTransition))
          cd = new AddFsmTransitionDialog(maaoha.MyTool.frame,
              moduleViews.get(i), null);
```

```
      else if (type.equals(Model.typeModule))
        cd = new UseModuleDialog(maaoha.MyTool.frame, moduleViews
            .get(i), null);

      if (cd == null)
        throw new MaaohaException(
            "ModelView->addNewGraphElement::Unknown type.");
      cd.pack();
      cd.setVisible(true);
      i = moduleViews.size();
    }
}

/*
 * Verifies model
 */
public void checkModel() throws MaaohaException {
  if (parent.getVerifyModel())
    for (int i = 0; i < moduleViews.size(); i++) {
      moduleViews.get(i).checkModule();
    }
}

/*
 * Collapses datapath cells
 */
public void colapseDatapathCells(Component comp) {
  for (int i = 0; i < moduleViews.size(); i++)
    if (moduleViews.get(i).getView() == comp) {
      moduleViews.get(i).getDatapath().colapseGraphCells();
    }
}

/*
 * Creates module
 */
public void createModule(String name) throws MaaohaException {
  Module module = new Module(name, model);
  model.addModule(module);
  ModuleView moduleView = new ModuleView(module, this);
  moduleViews.add(moduleView);

  ((DefaultTreeModel) parent.getTree().getModel()).insertNodeInto(
      moduleView.createTree(), treeNode, treeNode.getChildCount());

  parent.addModuleViewToModulesPane(moduleView);
  parent.getToolBar().updateCurrentModuleCombo();
}

/*
 * Creates tree node
 */
public DefaultMutableTreeNode createTree() {
  treeNode = new DefaultMutableTreeNode("System");

  for (int i = 0; i < moduleViews.size(); i++) {
    treeNode.add(moduleViews.get(i).createTree());
  }

  return treeNode;
}

/*
 * Returns module view
 */
public ModuleView findModuleView(Module module) {
  for (int i = 0; i < moduleViews.size(); i++) {
    if (moduleViews.get(i).getModule() == module)
```

```
      return moduleViews.get(i);
    }
    return null;
  }

  /*
   * Returns module view by component
   */
  public ModuleView findModuleViewBySelectedComponent(Component comp) {
    for (int i = 0; i < moduleViews.size(); i++)
      if (moduleViews.get(i).getView() == comp)
        return moduleViews.get(i);

    return null;
  }

  /*
   * Returns datapath graph scale
   */
  public double getDatapathGraphScale(Component comp) {
    return findModuleViewBySelectedComponent(comp).getDatapathGraphScale();
  }

  /*
   * Returns controller graph scale
   */
  public double getFsmGraphScale(Component comp) {
    return findModuleViewBySelectedComponent(comp).fsm.getFsmGraphScale();
  }

  /*
   * Return index of module view
   */
  public int getIndexOfModuleView(ModuleView moduleView) {
    return moduleViews.indexOf(moduleView);
  }

  /*
   * returns index of module view
   */
  public int getIndexOfModuleViewByView(Component view) {
    for (int i = 0; i < moduleViews.size(); i++) {
      if (moduleViews.get(i).getView() == view)
        return i;
    }
    return -1;
  }

  /*
   * Returns model
   */
  public Model getModel() {
    return model;
  }

  /*
   * Returns number of module views
   */
  public int getModulesNo() {
    return moduleViews.size();
  }

  /*
   * returns i-th module view
   */
  public ModuleView getModuleView(int i) {
    return moduleViews.get(i);
  }
```

```
  /*
   * Returns tool view
   */
  public ToolView getToolView() {
    return parent;
  }

  /*
   * Populates module panes
   */
  public void populateModulesPane(JTabbedPane pane) throws MaaohaException {
    for (int i = 0; i < moduleViews.size(); i++) {
      JSplitPane graphsPane = moduleViews.get(i).createView();
      graphsPane.setDividerLocation(400);
      pane.addTab(moduleViews.get(i).getModule().getName(), graphsPane);
    }
  }

  /*
   * Refreshes tabbed pane
   */
  public void refreshTabbedPane(Component comp) {
    for (int i = 0; i < moduleViews.size(); i++) {
      moduleViews.get(i).refreshTabbedPane(comp);
    }
  }

  /*
   * Removes module
   */
  public void removeModule(ModuleView moduleView) throws MaaohaException {
    ((DefaultTreeModel) parent.getTree().getModel())
        .removeNodeFromParent(moduleView.getTreeNode());
    parent.removeModuleViewFromModulesPane(moduleView);
    model.removeModule(moduleView.getModule());
    for (int i = 0; i < moduleViews.size(); i++) {
      moduleViews.get(i).getDatapath().removeUseModulesForModule(
          moduleView);
    }
    moduleViews.remove(moduleView);
    parent.getToolBar().updateCurrentModuleCombo();
  }

  /*
   * Sets datpath graph scale
   */
  public void setDatapathGraphScale(Component comp, double scale,
      Point2D.Double pt) {
    findModuleViewBySelectedComponent(comp)
        .setDatapathGraphScale(scale, pt);
  }

  /*
   * Sets controller graph scale
   */
  public void setFsmGraphScale(Component comp, double scale, Point2D.Double pt) {
    findModuleViewBySelectedComponent(comp).setFsmGraphScale(scale, pt);
  }
}
```

B.7.9 ModuleView.java

```
package maaoha.view.model;

import java.awt.Component;
```

```
import java.awt.Point;
import java.awt.geom.Point2D;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.util.ArrayList;

import javax.swing.JScrollPane;
import javax.swing.JSplitPane;
import javax.swing.tree.DefaultMutableTreeNode;

import maaoha.MaaohaException;
import maaoha.model.DatapathElement;
import maaoha.model.Expression;
import maaoha.model.FsmState;
import maaoha.model.FsmTransition;
import maaoha.model.Model;
import maaoha.model.Module;
import maaoha.model.Sfg;
import maaoha.model.UseModule;
import maaoha.view.TreeInfo;

import org.jgraph.graph.DefaultGraphCell;

/*
 * Class responsible for visualization of module
 */
public class ModuleView {
  protected ArrayList<DefaultGraphCell> useModuleToMark;

  protected ArrayList<DefaultGraphCell> transitionsToMark;

  protected DefaultMutableTreeNode treeNode;

  protected Module module;

  protected DatapathView datapath;

  protected FsmView fsm;

  protected JSplitPane view;

  protected ModelView parent;

  /*
   * COnstructor
   */
  public ModuleView(Module module, ModelView parent) {
    this.module = module;
    this.parent = parent;

    datapath = new DatapathView(module.getDatapath(), this);
    fsm = new FsmView(module.getFsm(), this);

    useModuleToMark = new ArrayList<DefaultGraphCell>();
    transitionsToMark = new ArrayList<DefaultGraphCell>();
  }

  /*
   * Adds new element to datapath
   */
  public void addNewElement(Point pt, String name, String elementType,
      String valueType) throws MaaohaException {
    DatapathElement de = new DatapathElement(name, elementType, valueType);
    module.getDatapath().addDatapathElement(de);
    datapath.addNewDatapathElementView(de, pt);
  }

  /*
```

```
 * Adds new expression
 */
public void addNewExpression(Point point, String expressionText,
    SfgView selectedSfg) throws MaaohaException {
  ArrayList<Expression> expressionList = module.getDatapath()
      .addExpressionFromString(selectedSfg.getSfg(), expressionText);
  datapath.addNewExpressionsView(expressionList, selectedSfg, point);
}

/*
 * Adds new controller state
 */
public void addNewFsmState(Point pt, String name) throws MaaohaException {
  FsmState state = new FsmState(name);
  module.getFsm().addFsmState(state);
  fsm.addNewState(state, pt);
}

/*
 * Adds new controller transition
 */
public void addNewTransition(FsmStateView source, FsmStateView target,
    String conditions, String sfgs) throws MaaohaException {
  FsmTransition trans = module.addFsmTransition(source.getFsmState(),
      conditions, sfgs, target.getFsmState());
  fsm.addNewTransition(source, trans);
}

/*
 * Adds new sub-module
 */
public void addUseModule(Point point, String name, Module selectedModule,
    String parameterList) throws MaaohaException {
  UseModule um = module.getDatapath().addUseModule(name, selectedModule,
      parameterList);
  datapath.addUseModule(um, point);
}

/*
 * Verifies module
 */
public void checkModule() throws MaaohaException {
  fsm.markCellsValid(transitionsToMark);
  useModuleToMark = new ArrayList<DefaultGraphCell>();
  transitionsToMark = new ArrayList<DefaultGraphCell>();

  boolean[] outputDefined = new boolean[datapath.getDatapath()
      .getOutputsNo()];

  for (int i = 0; i < fsm.getFsmStateNo(); i++) {
    FsmStateView stateView = fsm.getFsmState(i);
    for (int j = 0; j < stateView.getTransitionViewNo(); j++) {
      FsmTransitionView transView = stateView.getTransitionView(j);
      ArrayList<Sfg> sfgs = transView.getTransition().getSfgs();

      // reset outputs table
      for (int jj = 0; jj < outputDefined.length; jj++)
        outputDefined[jj] = false;

      ArrayList<DefaultGraphCell> expressionsToMarkForTransition =
        new ArrayList<DefaultGraphCell>();
      ArrayList<DefaultGraphCell> useModulesToMarkForTransition =
        new ArrayList<DefaultGraphCell>();

      for (int k = 0; k < sfgs.size(); k++) {
        ArrayList<DefaultGraphCell> expressionsToMarkForSfg =
          new ArrayList<DefaultGraphCell>();
        ArrayList<DefaultGraphCell> useModulesToMarkForSfg =
```

```
  new ArrayList<DefaultGraphCell>();
Sfg sfg = sfgs.get(k);
for (int l = 0; l < sfg.getExpressionsNo(); l++) {
  Expression expression = sfg.getExpression(l);

  // expression to be checked
  DatapathElement de = expression.getLeftSideElement();

  // set output defined
  if (de != null
      && de.getElementType().equals(Model.typeOutput))
    outputDefined[getDatapath().getDatapath()
        .getOutputIndex(de)] = true;

  // compare to all next expressions
  boolean errorExpression = false;
  boolean internalSfgError = false;

  if (expression.allElementsDefined() == false) {
    internalSfgError = true;
    errorExpression = true;
  } else
    for (int m = 0; m < sfgs.size(); m++) {
      for (int n = 0; n < sfgs.get(m)
          .getExpressionsNo(); n++) {
        Expression expression2 = sfgs.get(m)
            .getExpression(n);
        if (expression != expression2) {
          if (de == expression2
              .getLeftSideElement()) {
            errorExpression = true;
            if (m == k)
              internalSfgError = true;
            n = sfgs.get(m).getExpressionsNo();
          }
        }
      }
      if (errorExpression)
        m = sfgs.size();
    }

  // if this expression illegal, add it together with the
  // whole transition
  if (errorExpression) {
    ExpressionView expressionView = datapath
        .findExpressionView(expression);
    if (expressionView == null)
      throw new MaaohaException(
          "ModuleView->checkModule::expressionView == null");
    if (!expressionsToMarkForTransition
        .contains(expressionView.getGraphCell()))
      expressionsToMarkForTransition
          .add(expressionView.getGraphCell());
    // if the same sfg add it also to sfg view
    if (internalSfgError)
      if (!expressionsToMarkForSfg
          .contains(expressionView.getGraphCell()))
        expressionsToMarkForSfg.add(expressionView
            .getGraphCell());
  }

  // check if expression collides with UseModule output
  for (int m = 0; m < datapath.getUseModuleViewNo(); m++) {
    UseModule um = datapath.getUseModuleView(m)
        .getUseModule();
    for (int n = 0; n < um.getParametersNo(); n++) {
      DatapathElement de2 = um.getParameter(n);
      if (de2 == de)
```

```
              if (um.getModule().getDatapath()
                  .getInputOutputType(n).equals(
                      Model.typeOutput)) {
                // add use module to mark list
                UseModuleView umv = datapath
                    .findUseModuleView(um);
                if (!useModulesToMarkForTransition
                    .contains(umv.getGraphCell()))
                  useModulesToMarkForTransition
                      .add(umv.getGraphCell());
                if (!useModulesToMarkForSfg
                    .contains(umv.getGraphCell()))
                  useModulesToMarkForSfg.add(umv
                      .getGraphCell());

                // add expression to mark list
                ExpressionView expressionView = datapath
                    .findExpressionView(expression);
                if (expressionView == null)
                  throw new MaaohaException(
                      "ModuleView->checkModule::expressionView == null");
                if (!expressionsToMarkForTransition
                    .contains(expressionView
                        .getGraphCell()))
                  expressionsToMarkForTransition
                      .add(expressionView
                          .getGraphCell());
                if (!expressionsToMarkForSfg
                    .contains(expressionView
                        .getGraphCell()))
                  expressionsToMarkForSfg
                      .add(expressionView
                          .getGraphCell());
              }
            }
          }
        }
        datapath.findSfgView(sfg).setInvalidExpressions(
            expressionsToMarkForSfg);
        datapath.findSfgView(sfg).setInvalidUseModules(
            useModulesToMarkForSfg);
      }

      ArrayList<DefaultGraphCell> outputsToMarkForTransition =
        new ArrayList<DefaultGraphCell>();

      // marks undefined outputs
      for (int jj = 0; jj < outputDefined.length; jj++)
        if (outputDefined[jj] == false) {
          DefaultGraphCell cell = datapath.findElementView(
              datapath.getDatapath().getOutputByIndex(jj))
              .getGraphCell();
          outputsToMarkForTransition.add(cell);
        }

      if (expressionsToMarkForTransition.size() > 0
          || outputsToMarkForTransition.size() > 0
          || useModuleToMark.size() > 0) {
        // add transition to mark list
        if (!transitionsToMark.contains(transView.getGraphCell()))
          transitionsToMark.add(transView.getGraphCell());
      }

      transView.setInvalidOutputs(outputsToMarkForTransition);
      transView.setInvalidExpressions(expressionsToMarkForTransition);
      transView.setInvalidUseModules(useModulesToMarkForTransition);
    }
  }
```

```
  // check whether UseModules collides with themselves
  for (int i = 0; i < datapath.getUseModuleViewNo(); i++) {
    UseModule um = datapath.getUseModuleView(i).getUseModule();
    for (int j = 0; j < um.getParametersNo(); j++) {
      if (um.getModule().getDatapath().getInputOutputType(j).equals(
          Model.typeOutput)) {
        for (int i2 = i + 1; i2 < datapath.getUseModuleViewNo(); i2++) {
          UseModule um2 = datapath.getUseModuleView(i2)
              .getUseModule();
          if (um != um2) {
            for (int j2 = 0; j2 < um2.getParametersNo(); j2++) {
              if (um2.getModule().getDatapath()
                  .getInputOutputType(j2).equals(
                      Model.typeOutput)) {
                if (um.getParameter(j) == um2
                    .getParameter(j2)) {
                  // add both modules to mark list
                  UseModuleView umv = datapath
                      .findUseModuleView(um);
                  if (!useModuleToMark.contains(umv
                      .getGraphCell()))
                    useModuleToMark.add(umv
                        .getGraphCell());
                  umv = datapath.findUseModuleView(um2);
                  if (!useModuleToMark.contains(umv
                      .getGraphCell()))
                    useModuleToMark.add(umv
                        .getGraphCell());
                }
              }
            }
          }
        }
      }
    }
  }

  for (int i = 0; i < datapath.getUseModuleViewNo(); i++) {
    UseModuleView umv = datapath.getUseModuleView(i);
    if (umv.getUseModule().areAllParametersDefined() == false) {
      if (!useModuleToMark.contains(umv.getGraphCell()))
        useModuleToMark.add(umv.getGraphCell());
    }
  }
  fsm.markCellsInvalid(transitionsToMark);
}

/*
 * Create tree node
 */
public DefaultMutableTreeNode createTree() {
  treeNode = new DefaultMutableTreeNode(new TreeInfo(module.getName(),
      this));
  treeNode.add(datapath.createTree());
  treeNode.add(fsm.createTree());
  return treeNode;
}

/*
 * Creates view
 */
public JSplitPane createView() throws MaaohaException {
  JScrollPane dpView = datapath.createView();
  JScrollPane fsmView = fsm.createView();
  view = new JSplitPane(JSplitPane.VERTICAL_SPLIT, dpView, fsmView);
  view.addPropertyChangeListener("lastDividerLocation",
      new PropertyChangeListener() {
```

```
        public void propertyChange(PropertyChangeEvent ev) {
          datapath.dpGraph.repaint();
          fsm.fsmGraph.repaint();
        }
      });
  return view;
}

/*
 * Returns datapath view
 */
public DatapathView getDatapath() {
  return datapath;
}

/*
 * Returns datapath graph scale
 */
public double getDatapathGraphScale() {
  return datapath.getDatapathGraphScale();
}

/*
 * Returns controller graph scale
 */
public double getFsmGraphScale() {
  return fsm.getFsmGraphScale();
}

/*
 * Returns controller view
 */
public FsmView getFsmV() {
  return fsm;
}

/*
 * Return module
 */
public Module getModule() {
  return module;
}

/*
 * Returns model view
 */
public ModelView getParent() {
  return parent;
}

/*
 * Returns tree node
 */
public DefaultMutableTreeNode getTreeNode() {
  return treeNode;
}

/*
 * Returns view
 */
public JSplitPane getView() {
  return view;
}

/*
 * Refreshes both graphs
 */
public void refreshBothGraphs() {
```

```
    datapath.dpGraph.repaint();
    fsm.fsmGraph.repaint();
  }

  /*
   * Refreshes tabbed pane
   */
  public void refreshTabbedPane(Component comp) {
    if (comp == view) {
      datapath.dpGraph.repaint();
      fsm.fsmGraph.repaint();
    }
  }

  /*
   * Removes sfg view
   */
  public void removeSfg(SfgView sfgView) {
    datapath.removeSfg(sfgView);
    fsm.removeSfg(sfgView);
  }

  /*
   * Sets datapath graph scale
   */
  public void setDatapathGraphScale(double scale, Point2D.Double pt) {
    datapath.setDatapathGraphScale(scale, pt);
  }

  /*
   * Sets controller graph scale
   */
  public void setFsmGraphScale(double scale, Point2D.Double pt) {
    fsm.setFsmGraphScale(scale, pt);
  }
}
```

B.7.10 SfgView.java

```
package maaoha.view.model;

import java.util.ArrayList;

import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;

import maaoha.model.Sfg;
import maaoha.view.TreeInfo;

import org.jgraph.graph.DefaultGraphCell;

/*
 * Class responsible for visualization of signal flow graph
 */
public class SfgView {
  protected ArrayList<DefaultGraphCell> invalidUseModules;

  protected ArrayList<DefaultGraphCell> invalidExpressions;

  protected DatapathView parent;

  protected DefaultMutableTreeNode treeNode;

  protected Sfg sfg;

  protected ArrayList<ExpressionView> expressions;
```

```
protected DefaultGraphCell cell;

/*
 * Constructor
 */
public SfgView(Sfg sfg, DatapathView parent) {
  this.parent = parent;
  this.sfg = sfg;
  expressions = new ArrayList<ExpressionView>();
  for (int i = 0; i < sfg.getExpressionsNo(); i++) {
    expressions.add(new ExpressionView(sfg.getExpression(i), this));
  }

  invalidExpressions = new ArrayList<DefaultGraphCell>();
  invalidUseModules = new ArrayList<DefaultGraphCell>();
}

/*
 * Creates tree node
 */
public DefaultMutableTreeNode createTree() {
  treeNode = new DefaultMutableTreeNode(new TreeInfo(sfg.getName(), this));

  for (int i = 0; i < expressions.size(); i++) {
    treeNode.add(expressions.get(i).createTree());
  }

  return treeNode;
}

/*
 * Returns all graph cells
 */
public ArrayList<DefaultGraphCell> getCells() {
  ArrayList<DefaultGraphCell> cells = new ArrayList<DefaultGraphCell>();

  for (int i = 0; i < parent.elements.size(); i++) {
    cells.add(parent.elements.get(i).getGraphCell());
  }
  for (int i = 0; i < expressions.size(); i++) {
    cells.add(expressions.get(i).getGraphCell());
  }
  return cells;
}

/*
 * Returns i-th expression view
 */
public ExpressionView getExpression(int i) {
  return expressions.get(i);
}

/*
 * Returns number of expression views
 */
public int getExpressionsNo() {
  return expressions.size();
}

/*
 * Returns invalid expression
 */
public ArrayList<DefaultGraphCell> getInvalidExpressions() {
  return invalidExpressions;
}

/*
 * Returns invalid use-modules
```

```
 */
public ArrayList<DefaultGraphCell> getInvalidUseModules() {
  return invalidUseModules;
}

/*
 * Returns parent daapath view
 */
public DatapathView getParent() {
  return parent;
}

/*
 * Returns sfg
 */
public Sfg getSfg() {
  return sfg;
}

/*
 * Returns tree node
 */
public DefaultMutableTreeNode getTreeNode() {
  return treeNode;
}

/*
 * Refreshes tree
 */
public DefaultMutableTreeNode refreshTree() {
  treeNode.removeAllChildren();
  for (int i = 0; i < expressions.size(); i++) {
    treeNode.add(expressions.get(i).createTree());
  }

  return treeNode;
}

/*
 * Removes expression
 */
public void removeExpression(ExpressionView expressionV) {
  ((DefaultTreeModel) parent.parent.parent.parent.getTree().getModel())
      .removeNodeFromParent(expressionV.getTreeNode());
  expressions.remove(expressionV);
  sfg.removeExpression(expressionV.getExpression());
}

/*
 * Sets invalid expressions
 */
public void setInvalidExpressions(
    ArrayList<DefaultGraphCell> invalidExpressions) {
  this.invalidExpressions = invalidExpressions;
}

/*
 * Sets invalid use-modules
 */
public void setInvalidUseModules(
    ArrayList<DefaultGraphCell> invalidUseModules) {
  this.invalidUseModules = invalidUseModules;
}

/*
 * Sets name
 */
public void setName(String name) {
```

```
    sfg.setName(name);
    treeNode.setUserObject(new TreeInfo(name, this));
  }

  /*
   * (non-Javadoc)
   *
   * @see java.lang.Object#toString()
   */
  public String toString() {
    return sfg.getName();
  }
}
```

B.7.11 UseModuleView.java

```
package maaoha.view.model;

import javax.swing.tree.DefaultMutableTreeNode;

import maaoha.model.UseModule;
import maaoha.view.TreeInfo;

import org.jgraph.graph.DefaultGraphCell;

/*
 * Class responsible for visualization of sub-module
 */
public class UseModuleView implements GraphCellView {
  protected DefaultMutableTreeNode treeNode;

  protected UseModule module;

  protected DefaultGraphCell cell;

  protected DatapathView parent;

  /*
   * Constructor
   */
  public UseModuleView(UseModule module, DatapathView parent) {
    this.parent = parent;
    this.module = module;
  }

  /*
   * Creates tree
   */
  public DefaultMutableTreeNode createTree() {
    treeNode = new DefaultMutableTreeNode(new TreeInfo(module.getModule()
        .getName()
        + "(" + module.parametersString() + ")", this));
    return treeNode;
  }

  /*
   * (non-Javadoc)
   *
   * @see maaoha.view.model.GraphCellView#getGraphCell()
   */
  public DefaultGraphCell getGraphCell() {
    return cell;
  }

  /*
   * (non-Javadoc)
   *
```

```
   * @see maaoha.view.model.GraphCellView#getModelObject()
   */
  public Object getModelObject() {
    return module;
  }

  /*
   * Returns parent datapath view
   */
  public DatapathView getParentDatapathView() {
    return parent;
  }

  /*
   * Returns tree node
   */
  public DefaultMutableTreeNode getTreeNode() {
    return treeNode;
  }

  /*
   * Returns sub-module
   */
  public UseModule getUseModule() {
    return module;
  }

  /*
   * Sets graph cell
   */
  public void setGraphCell(DefaultGraphCell cell) {
    this.cell = cell;
  }

  /*
   * (non-Javadoc)
   *
   * @see java.lang.Object#toString()
   */
  public String toString() {
    String label = module.getName();
    if (module.getParametersNo() > 0) {
      label += "(";
      for (int i = 0; i < module.getParametersNo(); i++) {
        if (module.getParameter(i) != null)
          label += module.getParameter(i).getName();
        else
          label += "???";
        if (i < module.getParametersNo() - 1)
          label += ",";
      }
      label += ")";
    }

    label += " : " + module.getModule().getName();
    return label;
  }
}
```

B.8 Package maaoha.view.serialization

B.8.1 EdgeData.java

```
package maaoha.view.serialization;

import java.awt.geom.Point2D;
```

```
import java.util.Map;

import maaoha.model.FsmTransition;
import maaoha.view.model.FsmTransitionView;

import org.jgraph.graph.GraphConstants;

/*
 * Auxiliary class for serialization of edge data
 */
public class EdgeData {
  protected FsmTransition transition;

  protected Point2D labelPosition;

  /*
   * Constructor
   */
  public EdgeData(FsmTransitionView transView) {
    transition = transView.getTransition();
    Map mapEv = transView.getGraphCell().getAttributes();
    labelPosition = GraphConstants.getLabelPosition(mapEv);
  }

  /*
   * Returns label position
   */
  public Point2D getLabelPosition() {
    return labelPosition;
  }

  /*
   * Returns transition
   */
  public Object getTranstion() {
    return transition;
  }
}
```

B.8.2 ElementData.java

```
package maaoha.view.serialization;

import java.awt.geom.Rectangle2D;
import java.util.Map;

import maaoha.view.model.GraphCellView;

import org.jgraph.graph.GraphConstants;

/*
 * Auxiliary class for serialization of vertex data
 */
public class ElementData {
  protected Object object;

  protected Rectangle2D bounds;

  /*
   * Constructor
   */
  public ElementData(GraphCellView cellView) {
    object = cellView.getModelObject();

    Map mapEv = cellView.getGraphCell().getAttributes();
    bounds = GraphConstants.getBounds(mapEv);
  }
```

```
  /*
   * Returns vertex bounds
   */
  public Rectangle2D getBounds() {
    return bounds;
  }

  /*
   * Returns object
   */
  public Object getObject() {
    return object;
  }
}
```

B.8.3 ModelData.java

```
package maaoha.view.serialization;

import java.awt.geom.Rectangle2D;
import java.util.Map;

import maaoha.view.model.GraphCellView;

import org.jgraph.graph.GraphConstants;

/*
 * Auxiliary class for serialization of vertex data
 */
public class ElementData {
  protected Object object;

  protected Rectangle2D bounds;

  /*
   * Constructor
   */
  public ElementData(GraphCellView cellView) {
    object = cellView.getModelObject();

    Map mapEv = cellView.getGraphCell().getAttributes();
    bounds = GraphConstants.getBounds(mapEv);
  }

  /*
   * Returns vertex bounds
   */
  public Rectangle2D getBounds() {
    return bounds;
  }

  /*
   * Returns object
   */
  public Object getObject() {
    return object;
  }
}
```

B.8.4 ModuleData.java

```
package maaoha.view.serialization;
```

```
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Map;

import maaoha.view.model.DatapathElementView;
import maaoha.view.model.DatapathView;
import maaoha.view.model.FsmStateView;
import maaoha.view.model.FsmTransitionView;
import maaoha.view.model.FsmView;
import maaoha.view.model.ModuleView;
import maaoha.view.model.UseModuleView;

import org.jgraph.graph.DefaultGraphCell;
import org.jgraph.graph.GraphConstants;

/*
 * Auxiliary class for serialization of module
 */
public class ModuleData {
  protected ArrayList<ElementData> datapathElements;

  protected ArrayList<ElementData> controllerElements;

  protected ArrayList<EdgeData> edgeLabelsElements;

  /*
   * Constructor
   */
  public ModuleData(ModuleView moduleV) {
    datapathElements = new ArrayList<ElementData>();
    controllerElements = new ArrayList<ElementData>();
    edgeLabelsElements = new ArrayList<EdgeData>();

    DatapathView datapathView = moduleV.getDatapath();

    for (int i = 0; i < datapathView.getElementViewNo(); i++) {
      datapathElements
          .add(new ElementData(datapathView.getElementView(i)));
    }

    for (int i = 0; i < datapathView.getUseModuleViewNo(); i++) {
      datapathElements.add(new ElementData(datapathView
          .getUseModuleView(i)));
    }

    FsmView fsmView = moduleV.getFsmV();

    for (int i = 0; i < fsmView.getFsmStateNo(); i++) {
      FsmStateView fsv = fsmView.getFsmState(i);
      controllerElements.add(new ElementData(fsv));

      for (int j = 0; j < fsv.getTransitionViewNo(); j++) {
        edgeLabelsElements.add(new EdgeData(fsv.getTransitionView(j)));
      }
    }
  }

  /*
   * Returns edge label position
   */
  protected Point2D findEdgeLabelPosition(FsmTransitionView transitionView) {
    for (int i = 0; i < edgeLabelsElements.size(); i++) {
      EdgeData ed = edgeLabelsElements.get(i);
      if (ed.getTranstion().equals(transitionView.getTransition()))
        return ed.getLabelPosition();
    }
```

```
    return null;
  }

  /*
   * Returns element bounds
   */
  protected Rectangle2D findElementBounds(ArrayList<ElementData> list,
      Object obj) {
    for (int i = 0; i < list.size(); i++) {
      ElementData ed = list.get(i);
      if (ed.getObject().equals(obj))
        return ed.getBounds();
    }
    return null;
  }

  /*
   * Sets position of graph cells
   */
  public void restoreGraphCellPosition(ModuleView moduleView) {
    DatapathView datapathView = moduleView.getDatapath();
    Map<DefaultGraphCell, Map> nested = new Hashtable<DefaultGraphCell, Map>();
    for (int i = 0; i < datapathView.getElementViewNo(); i++) {
      Map attrMap = new Hashtable();
      DatapathElementView dev = datapathView.getElementView(i);
      GraphConstants.setBounds(attrMap, findElementBounds(
          datapathElements, dev.getElement()));
      nested.put(dev.getGraphCell(), attrMap);
    }

    for (int i = 0; i < datapathView.getUseModuleViewNo(); i++) {
      Map attrMap = new Hashtable();
      UseModuleView umv = datapathView.getUseModuleView(i);
      GraphConstants.setBounds(attrMap, findElementBounds(
          datapathElements, umv.getUseModule()));
      nested.put(umv.getGraphCell(), attrMap);
    }

    datapathView.getGraph().getGraphLayoutCache().edit(nested, null, null,
        null);

    FsmView fsmView = moduleView.getFsmV();
    nested = new Hashtable<DefaultGraphCell, Map>();
    for (int i = 0; i < fsmView.getFsmStateNo(); i++) {
      Map attrMap = new Hashtable();
      FsmStateView fsv = fsmView.getFsmState(i);
      GraphConstants.setBounds(attrMap, findElementBounds(
          controllerElements, fsv.getFsmState()));
      nested.put(fsv.getGraphCell(), attrMap);

      for (int j = 0; j < fsv.getTransitionViewNo(); j++) {
        FsmTransitionView ftv = fsv.getTransitionView(j);
        Map attrMap2 = new Hashtable();
        GraphConstants.setLabelPosition(attrMap2,
            findEdgeLabelPosition(ftv));
        nested.put(ftv.getGraphCell(), attrMap2);
      }
    }

    fsmView.getGraph().getGraphLayoutCache().edit(nested, null, null, null);
  }
}
```