TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers

Pablo Abad, Pablo Prieto, Lucia Menezo, Adrian Colaso, Valentin Puente, Jose-Angel Gregorio

University of Cantabria
What makes a Simulation Tool better than others?

**Flexibility**
- Heterogeneous field, from supercomputers to CMP.
- Highly Configurable

**Accuracy Vs. Comp. Effort**
- Avoid slow simulations for first stages of research process.
- But provide accurate enough results at last stages.

**Ease-of-Use**
- Fast learning is essential.
- MAX: 1-day delay for user-mode

**TOPAZ**
- Interfaz to Full-System Simulation.
- Multithreaded simulation for massive number of routers.
- Simple & Complex models.
- Dynamic accuracy simulation.
- Many out-of-the-box components.
- Very modular code, easy to understand.
Outline

• Simulator Description
• Out-of-the-Box
• Utilization Examples
• Support & Collaboration
Main Features

- Evolution of SICOSYS

- Object-oriented Design
  - Implemented in C++
  - 100 classes / 50,000 lines of code
  - High portability (C++ standard compiler)

- Different levels of detail

- Support for parallel execution

Main Features

- Evolution of SICOSYS
- Object-oriented Design
- Different levels of detail
- Support for parallel execution
Using TOPAZ (Building)

```bash
./TPZSimul -s SIMUL_DETAILED
```

**TPZSimul.ini**

```xml
<RouterFile id="../sgm/Router.sgm" />
<NetworkFile id="../sgm/Network.sgm" />
<SimulationFile id="../sgm/Simula.sgm" />
```

**Network.sgm**

```xml
<TorusNetwork id="TORUS" sizeX=8 sizeY=8 router="DETAILED" delay=1>
<MeshNetwork id="MESH" sizeX=8 sizeY=8 router="DETAILED" delay=1>
```

**Router.sgm**

```xml
<Router id="DETAILED" inputs=5 outputs=5 bufferSize=64 bufferControl="CT" routingControl="ROUTING_ALG">
<Injector id="INJ">
<Consumer id="CONS">
<Buffer id="BUF1" type="X+" headerDelay=2>
<Buffer id="BUF2" type="X-" headerDelay=2>
<Buffer id="BUF5" type="Node" headerDelay=2>
<Routing id="RTG1" type="X+" headerDelay=1>
<Routing id="RTG5" type="Node" headerDelay=1>
<Crossbar id="XBAR" inputs="5" outputs="5" type="CT">
<Input id=1 type="X+">
<Output id=5 type="Node">
<Connection id="C01" source="INJ" destination="BUF5">
<Connection id="C20" source="RTG.1" destination="XBAR.1">
```

**Simula.sgm**

```xml
<Simulation id="SIMUL_DETAILED">
<Network id="TORUS">
<SimulationCycles id=1000000>
<DiscardTraffic id=10000>
<TrafficPattern id="MODAL" type="RANDOM">
<Load id=0.5>
<PacketLength id=2>
```
Using TOPAZ (Printing)

Throughput/Latency curves

Latency Histogram

Injection/Consumption/Link map

+ Gems (or Gem5)

+ Orion

Power Breakdown

Throughput/Latency evolution
Outline

- Simulator Description
- Out-of-the-Box
- Utilization Examples
- Support & Collaboration
# Out of the Box

## 1. Configuration Parameters

<table>
<thead>
<tr>
<th>Router</th>
<th>Flow Control</th>
<th>Topology</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer Size</td>
<td>Virtual Cut Through</td>
<td>Ring</td>
<td>Random</td>
</tr>
<tr>
<td>Buffer Delay</td>
<td>Bubble Flow Control</td>
<td>Mesh (2D &amp; 3D)</td>
<td>Bit-Reversal</td>
</tr>
<tr>
<td>Packet Size</td>
<td>Wormhole</td>
<td>Torus (2D &amp; 3D)</td>
<td>Perfect-Shuffle</td>
</tr>
<tr>
<td># Virtual Channels</td>
<td>Virtual Channel flow Control</td>
<td>Midimew (2D)</td>
<td>Transpose Matrix</td>
</tr>
<tr>
<td># Physical networks</td>
<td></td>
<td>Square Midimew (2D)</td>
<td>Tornado</td>
</tr>
<tr>
<td>Message Types</td>
<td></td>
<td></td>
<td>Hot-Spot</td>
</tr>
<tr>
<td>Router Pipeline</td>
<td></td>
<td></td>
<td>Local</td>
</tr>
<tr>
<td>Link Delay</td>
<td></td>
<td></td>
<td>Trace-Based</td>
</tr>
</tbody>
</table>
## 2. Available Routers

<table>
<thead>
<tr>
<th>Router</th>
<th>REF</th>
<th>Year</th>
<th>Level of Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic with VC (Dally)</td>
<td>[16][17]</td>
<td>2001</td>
<td>Complex &amp; simple</td>
</tr>
<tr>
<td>VCTM (Dally + MC Support)</td>
<td>[18]</td>
<td>2008</td>
<td>Complex &amp; simple</td>
</tr>
<tr>
<td>Rotary Router</td>
<td>[19]</td>
<td>2007</td>
<td>Complex</td>
</tr>
<tr>
<td>Bufferless Router</td>
<td>[21]</td>
<td>2010</td>
<td>Simple</td>
</tr>
<tr>
<td>Bidirectional Router</td>
<td>[22]</td>
<td>2009</td>
<td>Simple</td>
</tr>
<tr>
<td>Buffered Crossbar</td>
<td>[23]</td>
<td>1987</td>
<td>Complex</td>
</tr>
</tbody>
</table>
3. Integration with Full-System Simulation Tools


Gem5 simulator system: [http://gem5.org/Main_Page](http://gem5.org/Main_Page)
Outline

• Simulator Description
• Out-of-the-Box
• Utilization Examples
• Support & Collaboration
## Increasing Full-System simulation accuracy

### Main System Parameters

<table>
<thead>
<tr>
<th>System</th>
<th>Parameters</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores</td>
<td>16 Cores, @4GHz, OOO, 4-wide issue, 64-entry IW, 16 outstanding Mem. Req</td>
<td>L2</td>
</tr>
<tr>
<td></td>
<td>16 MB, SNUCA, Token(B) coherence protocol, 6 msg. dependence chain</td>
<td>Topology</td>
</tr>
<tr>
<td>L1</td>
<td>Independent I/D caches, 32KB, 4-way, 1 cycles</td>
<td>L2 Bank</td>
</tr>
<tr>
<td></td>
<td>1MB, 16-way, 5 cycles, pseudo LRU</td>
<td>Links</td>
</tr>
<tr>
<td>Memory</td>
<td>4GB, 320GB/s, 260 cycles</td>
<td>OS</td>
</tr>
<tr>
<td></td>
<td>Solaris 10</td>
<td></td>
</tr>
</tbody>
</table>

### Broadcast Coherence Protocol (Execution Time)

[Chart showing normalized execution time for different benchmarks and protocol configurations]
Increasing Full-System simulation accuracy

Simulation speed (cycles/second)

More Accuracy => Slower simulations
On average, Ruby is ≈ 2X faster
Improving Simulation Speed (I)

Simulation speed (cycles/second)

Adaptive Interface

Throughput

Normalized Cycles Simulated/second

M Cycles simulated

Integer Sort

RUBY

TOPAZ

normalized Execution Time

Normalized Cycles Simulated/second

Execution Time

Simulation speed (cycles/second)
Improving Simulation Speed (II)

Simulation speed (cycles/second)

2-Thread Simulation

T1

T2

Normalized Cycles Simulated/second

RUBY
TOPAZ_SIMPLE
TOPAZ_COMPLEX
(AI)TOPAZ_SIMPLE
(AI)TOPAZ_COMPLEX
(P)TOPAZ_COMPLEX
(P)TOPAZ_SIMPLE

Normalized Execution Time

T1

T2

RUBY Normalized Execution Time

astar
hmem
bm
omnet++
BT_A
CG_A
FT_W
IS_A
LU_A
MG_W
SP_A
UA_A
Apache
JBB
OLTP
Zeus
blackholes
canneal
fluidanimate
streamcluster
MEAN
Simulating thousand-node Networks

12-Core (Xeon E5645) server with 54GBytes of main memory.

- 3D Torus, Bubble Router (simple), similar to IBM Blue Gene.
- Multithreaded implementation takes advantage of multicore server
- Good speedup for 1 Million routers
Outline

- Simulator Description
- Out-of-the-Box
- Utilization Examples
- Support & Collaboration
Support & Collaboration

tpzsimul
TOPAZ Interconnection network simulator

code.google.com/p/tpzsimul

Overview

TOPAZ is a general-purpose interconnection network simulator that allows the modeling a wide variety of message routers with different tradeoffs between speed and precision. TOPAZ comes from NICOSYS simulator, which was originally conceived to obtain results are very close to those obtained by using HDL description of networks components by hardware simulators but at lower computational cost. In order to make the tool easily comprehensible, extensible and reusable, the design of the tool is object-oriented and its implementation is in C++ language. For the models provided, approximately 110 classes, distributed in about 50,000 lines of code have been necessary. The simulator has support for parallel execution using standard POSIX threads. The portability is very high and can be used in any UNIX platform with a C++ standard compiler.

Announcements

• [10/02/2012] Initial release publicly available (See repositories)

Documentation

- Getting Started: A quick guide to understand how to get the sources and compile the simulator.
- Simulator Configuration: How to configure the simulator and most common command line options
- SGML Configuration: A quick review of SGML configuration files. How to define Routers, Networks and Simulations
- Components Available: A list of the components available in the tool
- GEMS Integration: A small review of GEMS integration and a guide to compile GEMS with TOPAZ networks
- GEM5 Integration: A small review of GEMS integration and a guide to compile GEMS with TOPAZ networks

References

If you use this tool in your research, please cite the following paper:

Support & Collaboration

Getting Started

First Steps

Configuration

Out of the Box

GEMS Integration

GEMS Integration

Guide to Compile TOPAZ network simulator

Step 1: Get It

You will need a Mercurial client in order to do so. If you have little idea about DVCS, we suggest to take a look at the best Mercurial tutorial out there.

hg clone https://code.google.com/p/tpzsimmul/ tpzsimmul

Step 2: Compile It

The code is structured in 4 directories:

- src includes the implementation of all classes, i.e., .cpp
- inc includes the specification of all classes, i.e., .hpp
- sgm includes SGML samples of different router microarchitectures, networks, etc...
- mak is where the simulator has to be compiled
Thanks for your attention

Questions?
Using TOPAZ

BUILDING

RUNNING

PRINTING
Using TOPAZ (Building)

**Router.sgm**
- Router & Crossbar Ports
- Buffer Size
- Routing & Flow Control Policies

**Network.sgm**
Using TOPAZ (Building)

Router.sgm
- Router & Crossbar Ports
- Buffer Size
- Routing & Flow Control Policies

Network.sgm
- Network Size
- Network Topology
- Link Delay

Simula.sgm
- Traffic Pattern
- Message Size
- Simulation Cycles
Using TOPAZ (Running)

No need to re-compile
- Only need to add new configurations at sgml files.
- Each configuration identified by a tag at Simula.sgm. Option –s option at command line to choose a specific configuration.

Different Execution Modes:
- Run your simulation for XXX Cycles.
- Run your simulation until YYY Messages reach their destination.

Command Line Options:
- Many sgml parameters can be overwritten through command line options.
- Example:
- Useful for scripting.