Thermal Aware Delay Modelling

Andreas Thor Winther

Kongens Lyngby 2011
IMM-M.Sc-2011-73

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

IMM-M.Sc: ISSN 0909-3192

Summary

Due to large temperature variations in VLSI circuits and the linear relationship between metal resistance and temperature, the delay through wires of the same length can be different. Traditional thermal aware floorplanning algorithms use wire length to estimate delay and routability.

It is shown that using wire length as the evaluation metric does not always produce a floorplan with the shortest delay. A temperature dependent wire delay estimation method for thermal aware floorplanning algorithms which takes into account the thermal effects on wire delay is implemented. In addition, the congestion and reliability issues are discussed as they are closely related to routing and temperature.

The MCNC macro block benchmark circuits are used to evaluate the proposed method and the experiment results show that the reduction in wire delay can be quite significant, up to a 20% decrease for the *xerox* benchmark, with a return cost of 1.01% increased area and 1.02% increased wire length. In *ami49*, which has the largest number of blocks, the area is increased by 1.01%.

Congestion as well as reliability worsen in some cases potentially creating congestion hotspots due to wires being re-routed through colder areas. To address this problem congestion and reliability are added to the evaluation metric thus effectively alleviating this problem.

Resumé

Grundet store temperatursvingninger i VLSI kredsløb og på grund af den lineære relation mellem metals modstand og temperatur kan signalforsinkelsen igennem ledninger af forskellig længde være forskellig. Traditionelle termisk-opmærksomme grundplanlægningsalgoritmer anvender længden af ledningen til at beregne signalforsinkelse og routeplanlægsmuligheder.

I afhandlingen vises, at brug af ledningslængde som vurderingsparameter ikke altid fører til en grundplan med den mindst mulige signalforsinkelse. I stedet implementeres en metode til beregning af en temperaturafhængig signalforsinkelse der endvidere inddrager den termiske effekt på forsinkelsen. I tillæg hertil diskuteres spørgsmål vedrørende ledningernes forstoppelse og pålidelighed, da de er tæt forbundne med rutefastlæggelse og temperatur.

Til at evaluere den foreslåede metode anvendes MCNC makro blok benchmark kredsløb. Forsøgsresultaterne viste, at reduktionen i signalforsinkelsen var ganske betydelig, op mod en 20% reducering for *xerox* kredsløbet, med et overhead på kun 1.01% større areal og 1.02% forøget ledningslængde. I *ami49*, som har det største antal blokke, er arealet forøget med 1.01%.

Ledningernes forstoppelse, såvel som pålidelighed, forværres i nogle tilfælde med risiko for opståen af hotspots som en følge af ruteplanlægning af ledninger gennem koldere områder. For at imødegå dette problem blev forstoppelse og pålidelighed inddraget i beregningsmålene, hvilket effektivt dæmpede problemet.

Preface

This thesis was prepared at Department of Informatics and Mathematical Modelling, at Danmarks Tekniske Universitet in partial fulfillment of the requirements for acquiring the M.Sc. degree in engineering. Part of the work behind this thesis was done at Arizona State University, USA.

The thesis deals with thermal aware wire delay estimation in floorplanning. Since congestion and reliability is closely related it is also examined. The main focus is on implementing and studying a new thermal aware model for wire delay but also models for congestion and reliability have been derived, implemented, and studied. The models have been implemented using C coding into HotFloorplan which is a tool used to create thermal aware floorplans.

A paper (Appendix E) for which I provided the results is included. The paper is based on the models and results implemented and found during the work of this thesis in the period from February 2011 - August 2011. The paper can be seen as a 4-page summary of the work done in this thesis.

Lyngby, August 2011

Andreas Thor Winther

Papers included in the thesis

[Appendix E] Wei Liu, Andreas Thor Winther, Alberto Nannarelli, Sarma Vrudhula *"Temperature Dependent Wire Delay Estimation in Floorplanning,"* August 2011. Accepted for publishing and presented at the 29th Norchip Conference November 2011 in Lund, Sweden.

Acknowledgements

I would like to thank my supervisor at Danmarks Tekniske Universitet, Prof. Dr. Alberto Nannarelli, for giving me the opportunity and support to write this project and for helping me to find a supervisor in Arizona, which is a big part of the reason I was able to spend a semester abroad while writing this thesis.

I would also like to thank Wei Liu for being a great help during the work of this thesis and for quickly answering any questions I had. Further, I would like to thank my host and supervisor at Arizona State University, Prof. Dr. Sarma Vrudhula, for letting me be part of his research group, and for supplying everything I needed during my stay at ASU.

Last but not least, I would like to thank all my friends and family for their invaluable work in proofreading and commenting the thesis.

Contents

Chapter 1

Introduction

With technology scaling, the feature sizes of both CMOS devices and wires shrink, effectively making it possible for designers to integrate even more functionalities into a single chip. This leads to a decrease in delay in CMOS transistors with every new process as the channel length is reduced. For metal wires, however, this is not always the case:

- For **local wires** ($< 100\ \mu$m), the distance between the end points become smaller due to scaling which further leads to a decrease in the delay.
- For **global wires**, which have to span across the chip, the delay increases due to the fact that the die size of the chip does not shrink but actually slightly increases with each new process.

In fact, the delay in global wires has increased steadily with technology scaling over the years and already dominate path delays.

In addition to technology scaling, the modelling of global wires is further complicated by thermal effects as described in Section 2.1. Due to the high degree of integration and various aggressive power management techniques (such as clock gating, power gating, voltage islands, etc.), the power consumption in different regions of the chip (e.g. the power density) can vary significantly. As

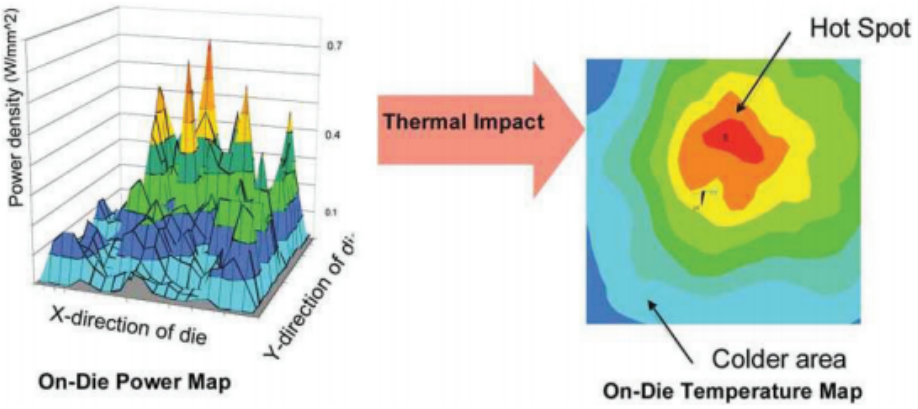


Figure 1.1: A typical power map and the corresponding thermal map (from [1])

illustrated in Figure 1.1, a high power density leads to a high thermal impact which again leads to thermal hotspots (areas with excessive heat). According to ITRS[2], the junction temperature of a semiconductor device must be kept at 85 °C or lower to ensure long term reliability and usually when temperatures reach 85 °C we define it as a hotspot. Consequently, the spatial non uniform power consumption within the chip exhibits as thermal gradients [1].

In recent years, temperature variation induced clock skew in clock distribution network has received much attention. In [3, 4], the authors described design time clock tree synthesis algorithms to modify merging locations against a nonuniform substrate thermal profile. While in [5], optimal insertion of tunable delay buffers into clock trees is discussed to adjust at run time the delay of clock distribution paths that are more susceptible to temperature variations. Thermal aware global routing algorithms for improving reliability are also discussed in [6, 7]. Even so, although extensive work has been done on thermal aware floorplanning regarding global signal wires all of them assume electrical resistivity in wires to be constant and that thermal gradients in the substrate has no impact on wire delay. This assumption is in general invalid and increasingly inaccurate in nanometer high performance designs where large temperature gradients already exist in the substrate.

[1] A thermal gradient is the difference in temperature, $|T_A - T_B|$, between two blocks, A and B

1.1 Problem Description

Wire delay is temperature dependent since resistivity in metal increases with temperature. Global wires are routed over many functional blocks which exhibit a large thermal gradient along the length of the wire; this nonuniform thermal profile can significantly degrade wire performance. In nanometer technologies more and more circuits are limited by wire delay rather than gate delay, therefore optimization of wire delay is very important and sometimes crucial.

The aim of this project is to investigate floorplan and wireplan methods to mitigate the increasing temperature dependent delay in global wires. The optimization process shall consider a wire delay model, the wire temperature profile, and the actual shape of nets. As congestion and reliability are closely related these will be considered too. The optimization algorithm shall then be validated against a set of benchmark circuits.

This can be summarized into:

- Implement a global wire planning algorithm in HotFloorplan (L-shape routing only and no ripup, no reroute, which is simpler than global routing). Cost function in HotFloorplan modified to weighted sum of Tmax, Area and QoW (quality of wire), coding in C.
- Implement a temperature dependent wire delay model in HotFloorplan, coding in C.
- Compare the quality of solution in HotFloorplan in terms of: area, maximum substrate temperature, wire length, delay, congestion, and reliability.
  - Perform wire planning with traditional cost function (thermal UNAWARE delay, congestion, reliability).
  - Perform wire planning with new cost functions (thermal AWARE delay, congestion, reliability).

1.2 Tools Used

HotSpot v5.0

HotSpot is a tool developed by [8] and the computer science department at the University of Virginia [2]. Further information can be found at [9].

According to the homepage that HotSpot is released and maintained from [9]:

"HotSpot is an accurate and fast thermal model suitable for use in architectural studies. It is based on an equivalent circuit of thermal resistances and capacitances that correspond to microarchitecture blocks and essential aspects of the thermal package. The model has been validated using finite element simulation. HotSpot has a simple set of interfaces and hence can be integrated with most power-performance simulators like Wattch. The chief advantage of HotSpot is that it is compatible with the kinds of power/performance models used in the computer-architecture community, requiring no detailed design or synthesis description. HotSpot makes it possible to study thermal evolution over long periods of real, full-length applications."

In this thesis it is used to generate steady-state thermal maps of the floorplans outputted by HotFloorplan. (Chapter 3). In addition, HotFloorplan invokes function calls from HotSpot.

HotFloorplan

HotFloorplan is a thermal aware floorplanner that is part of the HotSpot package and is used to find the best floorplan when given a floorplan description file as input. It evaluates the *best* floorplan based on a cost function and this evaluation is a very central part of our work. This tool is thoroughly introduced in Chapter 3.

TexMaker 3.02, WinEdt 6 and MikTex 2.9

This report was made using LaTeX. MikTex 2.9 was used to compile the .tex files. TexMaker 3.02 and WinEdt 6 were both used to edit the .tex files. The

[2] http://www.cs.virginia.edu/

design template (preample) was provided by the Department of Informatics and Mathematical Modelling (IMM) under which this thesis is written.

InkScape, Adobe Illustrator, Adobe Photoshop

Inkscape is a program used to open .svg (scalable vector graphic) files which is the format used to store thermal- and congestion- maps. Adobe Illustrator and Adobe Photoshop were used for most of the remaining pictures in this report.

Putty, WinSCP and servers used to run simulations

During the entire timeline of this process I had servers available to run the simulations. When I was at Arizona State University I used the habanero server (run by Ira A. Fulton School of engineering) and when I was at Danmarks Tekniske Universitet i used the eda7 server (run by the Department of Informatics and Mathematical Modelling). To access these files and to run the simulations I used putty to create an SSH connection and WinSCP to transfer files between the server and my local computer.

1.3 Structure of the Report

In Chapter 2 some key topics relevant to this thesis are introduced. In Chapter 3 a detailed introduction to the main tool used in this report is given, including the underlying simulation method (Simulated Annealing) and how the floorplans are evaluated. In Chapter 4, the new models (thermal aware wire delay, congestion, and reliability) are derived and explained. In Chapter 5 the experiments performed in this thesis are clarified and the results are presented and commented. In Chapter 6 some important design choices as well as potential future work is discussed. Finally in Chapter 7 the report is concluded.

Chapter 2

Background

In this chapter some key topics needed to have a full understanding of the thesis are introduced. The main sources of heat and the thermal effects that these induce on VLSI ciruits are introduced. Then, a general introduction to circuit floorplanning is given. Finally, the routing scheme implemented in this work is presented.

2.1 Thermal Effects on VLSI Circuits

Power dissipated in a VLSI [1] circuit manifests in the form of heat. With MOSFETs [2], when a signal transition occurs, the applied voltage leads to a lateral electric field, which then accelerates the charge carriers to move from source to drain. The charge carriers gain kinetic energy and when they collide with other carriers or atoms, part of this energy is released causing vibrations of these particles, leading to a rise in temperature. In this way, electrical energy due to power consumption in a CMOS [3] circuit is transformed to heat.

[1] VLSI: Very Large Scale Integration
[2] MOSFET: Metal-Oxide-Semiconductor Field-Effect Transmitter
[3] CMOS: Complementary Metal-Oxide Semiconductor

The thermal effect is more prominent in global wires than in local wires for two main reasons:

. Global wires are routed in the top metal layers away from the heat sink resulting in higher temperatures.

. Global wires are both long and routed across many different parts of the chip which easily leads to large thermal gradients

According to [10], two sources of heat exist in a CMOS circuit: cells and interconnects (although, power consumption in cells is outside the scope of this thesis). Power consumption in interconnects, on the other hand, is caused by the flow of a small amount of current that charge and discharge the parasitic capacitance in the interconnect during signal transition. This small amount of current flow can potentially lead to a significant temperature increase in metal wires, widely known as self-heating, resulting from very low-κ dielectric materials used in modern processes [4].

Furthermore, according to [10], for a flip-chip design[5], also known as a Controlled Collapse Chip Connection (C4), heat generated from the transistor junctions dissipating to the ambient environment through the heat sink attached to the back side of the silicon substrate constitutes the primary heat conduction path. However (and for the work of this thesis more importantly), a secondary heat conduction path also exists from the substrate towards the packaging and printed circuit boards (PCB) through the metal layers.

In nanometer technologies, in spite of an increase in the number of available metal layers, the top metal layers may still get closer to the substrate which results in a stronger thermal coupling between the substrate and the wires. Figure 2.1 shows a typical flip chip design with a cross-sectional view of the PCB, packaging, and a heat sink. The heat sink is attached to the backside of the substrate through thermal interface materials (TIM) and the wire leads are connected to the PCB through the C4. The main transfer path is due to conduction within the system and through radiation to the ambient.

Reliability is another factor also widely affected by temperature [13]. The mean time to failure (MTTF) for devices, which directly impacts the overall system reliability, is reduced with higher temperature. In [14] it is reported that a small

[4] low-κ dielectrics refer to semi-conductors with a small dielectric constant relative to silicon dioxide [11]

[5] Flip-chip, is when the surface of the chip is covered with an array of pads on the top of the metal. Because the chip is flipped up-side down it is in very close proximity to the package, eliminating the inductance associated with the bond wires[11]

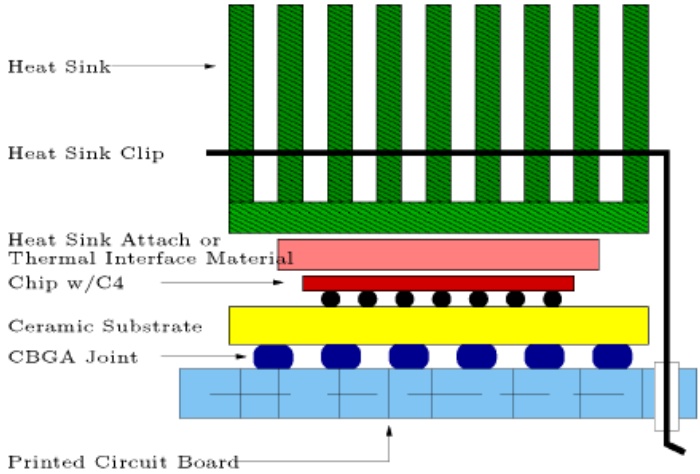


Figure 2.1: Heat dissipation paths of a chip (from [12])

increase in operating temperature (from 10 → 15C) can reduce the lifespan of devices by a factor 2. Many physical effects that cause a reliability degradation are thermally activated processes ([15, 16, 17]). Failure mechanisms such as electromigration are accelerated by high temperatures or temperature gradients and can cause permanent device failures [18].

To conclude, high temperatures and large thermal gradients in the metal layers can affect many aspects of interconnect design, including signal delay, routing congestion and reliability. The propagation delay in metal wires is severely degraded by high temperature as the electrical resistivity in metal increases linearly with temperature. Furthermore, these large within-die thermal gradients can lead to unpredictability and performance mismatch between wires of the same length, as it is not feasible to assume that two wires that are subject to different temperatures perform alike.

Traditional physical design algorithms such as floorplanning and routing assume resistivity in interconnects to be uniform and constant which is why wire length often is used as a metric to estimate signal delay and congestion of interconnects. Due to the reasons just mentioned, for designs where the substrate has a nonuniform thermal profile, the traditional way of estimating wire delay can lead to significant errors.

2.2 Floorplanning

Physcial design begins with a floorplan which to a large extent determines the quality of the final design. In microelectronics, as soon as you have some loosely defined logic, an initial floorplan is prepared to see if the logic will fit in the chip. This is a very iterative process as the floorplan will often suggest changes to the logic (and microarchitecture), which in turn changes the floorplan. Traditional floorplanning algorithms only optimize the total area and wire length. The floorplanner estimates the area of major units in the chip and defines their relative position. These are eventually defined as macro blocks (functional blocks). An example floorplan is given in Figure 2.2.

During recent years, as thermal issues have become more prominent, this has become increasingly misleading and to meet this problem peak temperature has been added to the cost functions in what is known as thermal aware floorplanning algorithms [19, 20, 21]. Thermal coupling between high power consumption blocks can significantly affect the temperature distribution in the whole chip which is why thermal aware floorplanning algorithms consider peak temperature in the evaluation of a floorplan (in addition to area and wire length).

To evaluate the methods proposed in this thesis we use a set of benchmarks that include a floorplan description file. This file holds information about the macro blocks in the circuit, ready for a floorplanning tool to produce a final floorplan. HotFloorplan is such a thermal aware floorplanner and the work in this thesis is based on improving this tool by implementing a more advanced thermal aware model.

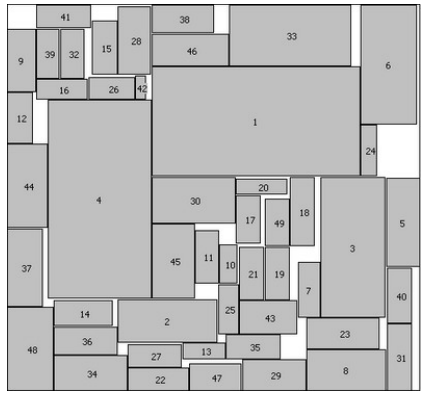


Figure 2.2: Example Floorplan. The grey boxes represent macro blocks while the white boxes is empty unused space, also known as white space (from [22]).

2.3 MCNC Benchmark suite

The Microelectronics Center of North Carolina (MCNC) design automation benchmarks (more formally referred to as the *International Symposium on Circuits and Systems (ISCAS)* benchmarks or *The Association for Computing Machinery, Design Automation Special Interest Group (ACM/SIGDA)*) is a collection of benchmarks contributed by organizations and individual members over a period of years. More specifically we are using the building block (macro block) place & route benchmarks in Table 2.1:

Benchmark	# Blocks	Nets
hp	11	83
apte	9	97
ami49	49	408
xerox	10	203
ami33	33	123

Table 2.1: MCNC macro block benchmark overview

The first set originated at the Special Session of ISCAS '85, where 10 research teams were brought together to present their experimental results with combinational test generation algorithms. This trend continued for the following decade providing a number of benchmarks representing a wide variety of problem domains - the macro block benchmarks used in this thesis being some of them - making these benchmarks by far the most frequently cited of all benchmarks [23].

Each benchmark comes with a floorplan description file which only contains information about the names, total area and the min/max aspect ratio of the functional blocks in a given circuit. It does not give any information about where exactly the functional blocks are on the chip or what the aspect ratio of the chip is. It is these files we use in this thesis.

2.4 Routing

Detailed routing information of an interconnect is unknown at the floorplanning stage and wirelength is usually estimated as the Manhattan distance[6] between two connected blocks. The half perimeter can be along the lower bend or the

[6]Manhattan distance: Defined as the half perimeter of the bounding box of two end points of an interconnect

upper bend and when thermal effects are not considered, the resistance is constant along the interconnect meaning that either route would have the same delay.

For thermal aware wire delay estimation, however, different routes of the same length can have very different delay since they are subject to different thermal profiles.

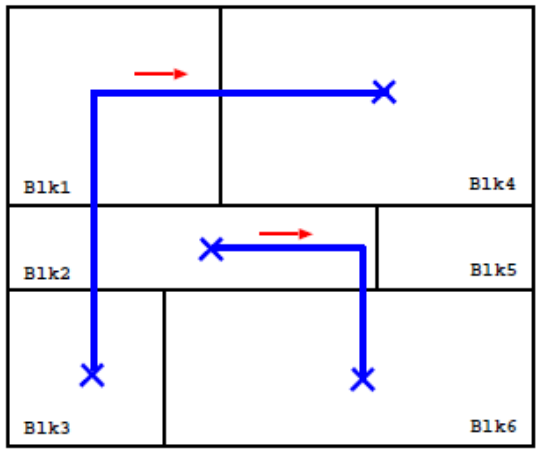


Figure 2.3: We use only upper bend L-shape routing with signals propagating from left to right. (from [24])

In this thesis, the routing of interconnects are restricted to upper bend L-shape routing as illustrated in Figure 2.3. L-shape routing, or 1-bend routing, uses a predefined pattern (in this case an L-shape) to route two end points. This type of routing is known as pattern routing. Pattern routing can reduce the number of vias and, according to [25], allows a more accurate prediction of wirelength and congestion at an early stage in the design flow. We also assume all signals propagate from left to right between two end-points. Furthermore, we assume that all wires start and end in the center of the start and end block respectively. Finally, no rip-up or re-routing is performed leaving the routing very simple.

In fact, only two possible routes between two end points exists. These are either up/right or right/down upper bend routing and considering the restrictions just described, only one of those routes are legal for any given endpoints meaning that no actual design choice is made when routing the wires. If the start point is lower than the end point (like the path from Blk3 $\rightarrow$ Blk4) only an up/right upper bend route is possible and when the start point is higher than the end point (like the path from Blk2 $\rightarrow$ Blk6) only a right/down upper bend route is possible.

Chapter 3

HotFloorplan - Method and Evaluation

In this chapter HotFloorplan (HFP), the main tool used in this project, is introduced. It is an architectural level thermal aware floorplanning tool, is part of the HotSpot package [9], and is developed and released by the same research team as HotSpot [8].

This chapter contains an overview of the simulation process, to give a general idea of how HFP works, and briefly describes the methods behind it. Then, in Section 3.2 and Section 3.3, the two key concepts used by HFP: *simulated annealing* and *floorplan evaluation* are fully introduced. Finally, because the simulation process relies on a number of parameters it is described how to find and set these.

Initially HFP uses Classic Simulated annealing and a basic cost function (Cost Function 1) to evaluate floorplans. In this thesis Fast-Simulated Annealing has been implemented to speed up the simulation process and two new cost functions (Cost Function 2 and 3) are added. Cost Function 2 includes an improved thermal aware wire delay model, while Cost Function 3 includes congestion and reliability.

3.1 Simulation Overview

The purpose of HotFloorplan (HFP) is to find the best floorplan when given a floorplan description file as input. A floorplan description file only contains information about the names, total area and the min/max aspect ratio of the functional blocks in a given circuit. It does not give any information about where exactly the functional blocks are on the chip or what the aspect ratio of the chip is.

The pseudocode for the simulation is given in Listing 3.1.

Listing 3.1: Pseudocode for the HotFloorPlan Algorithm

```
Set initial temperature;
step = 0;
try = 0;
while (steps != max steps && probability > minimum) {
  create initial floorplan;
  while (try < max tries) {
    try++;
    move block around randomly according to probability;
    evaluate floorplan;
    if (current floorplan > best floorplan)
      best floorplan = current floorplan;
  }
  step++;
  change simulated annealing temperature;
  calculate probability as a function of temperature;
}
output best floorplan;
```

HotFloorplan uses simulated annealing which, generally speaking, is an optimization process used to find the global optimum in a solution space. Translated to something relatable to this thesis, it is used to find the best floorplan within a set of constraints (block size etc.). Out-of-the-box, HFP uses classic simulated annealing. However, in this thesis it has been modified to use Fast-Simulated Annealing (Fast-SA) which is a faster type of simulated annealing. In brief, the simulation goes through a series of steps and with every step a synthetic *temperature* is changed according to an annealing schedule. The temperature is used to define how wide the search for a better floorplan is (within the solution space) - a low temperature means a narrow search. A detailed introduction to both classic annealing and Fast-SA is given in Section 3.2.

Within each step, HFP starts out with an initial floorplan that is simply any possible floorplan within the solution space. HFP then tries to optimize this floorplan by moving the blocks around (small moves if the temperature is low), and for every move it evaluates the floorplan using a cost function. If a new floorplan is better it gets accepted and if it is worse, it gets rejected. In the

end we get the best floorplan available at that temperature. Next step repeats this action only this time with another annealing temperature according to the annealing schedule. In the end we get the best floorplan created during all steps. The algorithm either ends when the maximum number of steps has been reached or if the probability (which is a function of temperature) of the next step is under a certain value. How HFP decides which floorplan is *best* is vital to this thesis as it is this evaluation process that has been modified to include thermal awareness. Detailed information on the evaluation process is given in Section 3.3.

Unaltered, HotFloorplan makes 7000 tries with each step, and has a hard cap of 1000 steps. In our modified version, 7000 tries are still made with each step but the hard cap of steps has been changed to 50 since Fast-SA has been implemented.

3.2 Simulated annealing

A problem often faced when dealing with optimization processes is that due to certain constraints(time, processor availability, number of objects, etc.), finding the global optimum becomes unmanageable. Simulated annealing helps alleviate this problem. In the following two sections we will first introduce the classic form of simulated annealing and then follow with a faster method called Fast-Simulated Annealing (Fast-SA).

3.2.1 Classic Simulated Annealing

To illustrate the usefulness of simulated annealing [1] we use the travelling salesman problem [2]. One way to solve this problem (hence referred to as *the simple approach*) is to start out with a random itinerary and pairwise trade the order of visits to the cities. After every pairwise trade, the total distance is evaluated and if it is shorter than the current shortest distance, a new shortest route has been found. The problem with this approach is that even though it quickly finds a local optimum, it is unable to proceed further to find the global optimum. Simulated annealing (SA) improves this approach in two distinct ways.

[1] Simulated annealing: Named so after the process undergone by misplaced atoms in a metal when its heated and then slowly cooled

[2] The travelling salesman problem is the problem of finding a travel route through a large number of cities while keeping the distance that the salesman has to travel as short as possible

The first is that while the simple algorithm strictly chooses routes that give a better *immediate* optimum, known as a *downhill* move, SA sometimes chooses a **worse** immediate optimum based on the presumption that this will later result in a **better** *final* global optimum, known as an *uphill* move. If the probability that choosing a worse immediate optimum results in a better final optimum is high enough, the trade is made. In this way, with SA, the problem of only finding local optima can be avoided.

This is also known as the *Metropolis algorithm*[26] in which the criterion for allowing an uphill trade is:

$$e^{\frac{-\Delta D}{T}} > R(0,1) \tag{3.1}$$

where ΔD is the change in distance resulting from the trade (negative for a downhill trade; positive for an uphill trade), T is a synthetic "temperature", and R(0,1) is a random number between 0 and 1. D is a cost function and in the original case of annealing a metal it represents the free energy. The equation shows that with a larger T, more uphill moves are allowed and a much larger part of the solution space is accessed. Finally, it shows that objects subject to trading are generally chosen randomly.

The second is scheduling of the temperature, T. As previously described, a larger T allows for more uphill moves and after a certain amount of moves or when the cost function starts to decline slowly, the temperature is lowered. This reduces the search space and the amount of allowed uphill moves. The temperature is lowered many times effectively "quenching" the process; in the end only allowing downhill moves so as to find the local optimum.

In this thesis, using the travelling salesman analogy, the cities are the functional blocks in the floorplan and the cost function is based on the models implemented during the underlying work of this thesis (area, maximum substrate temperature, wire delay, congestion, reliability) as described in Chapter 4. The temperature, T, declines to 0.99 of the previous value with every step and stops either when the probability of the next step resulting in a better solution is too low or if a hard cap of 1000 steps has been reached. These values can easily be changed through a configuration file. The output of hotfloorplan shows current step number, current temperature, the number of tries (pairwise trades), the number of accepts, the number of rejects, the average cost (based on the cost function) of the current step and the best cost of this and all previous steps [27, 28, 29].

3.2.2 Fast-Simulated Annealing

As mentioned earlier, Fast-Simulated Annealing (Fast-SA), is a method that seeks to improve the speed of the simulations quite significantly. As an example, the slowest MCNC benchmark circuit, ami49, takes about 8-9 hours to simulate using classic SA while it only takes 40-50 minutes using Fast-SA. The upside of Fast-SA is the increased speed, while the downside is that some floorplans might not be tried thus decreasing the chance of finding the best possible floorplan. While classic SA only reduces the temperature at a fixed rate throughout the entire simulation, Fast-SA goes through 3 stages and relies on a number of parameters (c, k and Δ_{avg}) to define the schedule and tweak it before running the simulation. Careful setting of these parameters ensures results that are very close to those of classic SA and in that way effectively reduces the downside of Fast-SA [30].

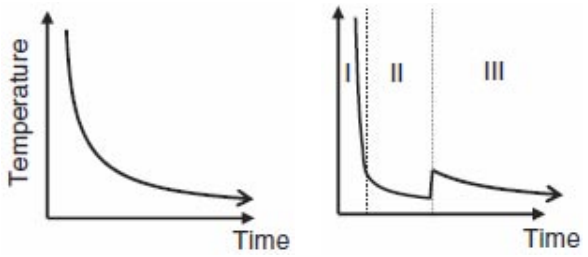
According to [30] it is neither effective nor efficient to accept too many uphill moves in the beginning of the simulation and the motivation behind Fast-SA is to reduce these. Therefore, a greedy search algorithm is applied to find a local optimum faster. As soon as the local optimum is found, we continue with classic SA.

To implement this scheme, Fast-SA can be said to consist of three stages:

1) A high temperature random search stage
2) A pseudo-greedy local search stage
3) A hill climbing stage

In Stage 1, the temperature is set very high, meaning that the probability of accepting an inferior solution (uphill climb) approaches 1. This is to avoid getting trapped at a local optimum in the very beginning. This stage is usually only one step.
In Stage 2, it is the opposite with a temperature approaching zero to accept only a small number up uphill climbs.
In Stage 3, the temperature is raised to allow hill climbing again to search for better solutions. The temperature is gradually declining until it is very likely to converge to a desired solution. The temperature over search time for classic SA and Fast-SA are shown in Figure 3.1a and b respectively.

In this way, Fast-SA saves significant steps to explore solution space and is able to devote more time to finding better solutions in stage 3 (hillclimbing), making it more efficient. To give an idea of the increase, with classic SA and our version of HFP, 4-600 steps are usually needed to finish the simulation and a hard cap



(a) Classic SA (b) Fast-SA

Figure 3.1: Temperature scheduling (temperature over time) for (a) classic SA and (b) Fast-SA (from [30])

of 1000 steps was set while with Fast-SA no more than 50 steps are needed. To implement Fast-SA the function for the temperature, T, can be derived by the following equations:

$$T_{step} = \begin{cases} \frac{-\Delta_{avg}}{\ln(P)} & step = 1 \\ \frac{T_1}{step \cdot c} & 2 \leq step \leq k \\ \frac{T_1}{step} & step > k \end{cases} \tag{3.2}$$

where *step* is the current iteration, Δ_{avg} is the average uphill cost, P is the initial probability for accepting uphill solutions, and c and k are the tunable user-defined parameters mentioned earlier. The first iteration is stage 1 where the temperature is set according to P (initial probability) and Δ_{avg} (average uphill cost) and because P is set close to 1, it performs a random search to find a good solution. Then it continues into stage 2 where it performs a pseudo-greedy search until the kth iteration, with c being the controlling parameter for how low the temperature is - a large c is chosen to make $T \to 0$ which ensures that only good solutions to perform pseudo-greedy searches are accepted. Finally, in stage 3, after k iterations, the temperature increases again to further improve solution quality and the scheme continues like classic SA until the final solution has been found. Note that the initial temperature is the same for both schemes, namely $T_1 = -\Delta_{avg}/\ln(P)$ and in both cases has to be kept high to avoid getting trapped at a local optimum in the very beginning.

3.3 Floorplan Evaluation

As mentioned in Section 3.1, HotFloorplan (HFP) relies on a cost function to evaluate how good a floorplan is. In this thesis we are comparing three cost functions: CF1, CF2 and CF3. CF1 is the original HFP cost function, CF2 is an improved cost function where the wire delay model is now thermal aware, while CF3 is CF2 with congestion and reliability added to its cost. CF1 and CF2 are the main cost functions to be compared, while CF3 is chosen because congestion and reliability is closely related and examining the effects of an added focus on these two metrics provides a number of interesting aspects. In this section, all three cost functions will be introduced. A detailed introduction of the new models (wire delay, congestion, reliability) can be found in Chapter 4. With all three cost functions, further information on how to find and choose the parameters that these cost functions rely on is given in Section 3.4.1.

3.3.1 Cost Function 1: Original HotFloorplan Evaluation

Cost Function 1 (CF1) is the original HotFloorplan cost function. It is based on 3 metrics: **area** is the total area of the chip (height · width), **Tmax** is the maximum substrate temperature, and **wire_length** is the total wire length of all the wires in the floorplan. The model only includes very simple models and wire delay is not included directly in the evaluation. From the package, however, HFP *does* calculate the wire delay but simply as wire length multiplied with a constant - meaning that wire delay is proportional to wire length and it does not vary with neither substrate temperature nor wire temperature. Three parameters are used to weigh the three metrics, and the higher value of the parameter, the higher the importance of the metric. The three parameters are λ_A, λ_T, and λ_W and the final cost function is given in Equation (3.3) below:

$$CF1 = \lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_length \tag{3.3}$$

3.3.2 Cost Function 2: Improved Wire Delay Model

Cost Function 2 (CF2) has a modified wire delay model. This cost function still uses **area** and **Tmax** as its first two metrics but instead of simply using wire length as the third, it uses **wire_delay** which is a function of both wire length, electrical resistance, substrate temperature, internal wire temperature and a number of technology constants. The final delay model, based on [31], is derived and introduced in Section 4.2, and the most important results, Equation (4.1), (4.6), and (4.7), are repeated below:

The final interconnect wire delay, D, is:

$$D = D_0 + (c_0 L + C_L) r_0 \beta \int_0^L T(x)\, dx - c_0 r_0 \beta \int_0^L x \cdot T(x)\, dx$$

where

$$D_0 = R_0(c_0 L + C_L) + \left(c_0 r_0 \frac{L^2}{2} + r_0 L C_L \right)$$

D_0 is the Elmore delay of the interconnect corresponding to the unit length resistance at 300K (reference temperature), c_0 and r_0 is the sheet capacitance and resistivity respectively, L is the total length of the wire, C_L is the load capacitance, T(x) is the temperature of the wire at position x, β is the temperature coefficient and R_0 is the driver cell's ON resistance. The temperature along the length of the wire at position x is:

$$T(x) = T_{sub} + \frac{\theta}{\lambda^2} \cdot \left(1 - \frac{\sinh \lambda x + \sinh \lambda (L - x)}{\sinh \lambda x} \right), 0 < x < L$$

where the constants θ and λ are dependent on the chosen metal layer in a specific technology node. T_{sub} is the temperature of the substrate, and L is the total length of the wire.

As with CF1, we have three parameters used to weigh the three metrics and as in CF1 the three parameters are λ_A, λ_T, and λ_W which gives us a final cost function given in Equation (3.4) below:

$$CF2 = \lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_delay \tag{3.4}$$

3.3.3 Cost Function 3: Added Congestion and Reliability Models

Cost Function 3 (CF3) is CF2 with two extra metrics added: congestion and reliability. As CF2 it uses **area**, **Tmax**, and **wire_delay** but CF3 further utilizes a congestion model and a reliability model which are further described in Section 4.3 and Section 4.4 respectively. We will here give a brief summary of these models.

The congestion model is based on a congestion map made up of a two dimensional matrix, which divides the floorplan into a 80μm $\cdot$ 80μm cell grid map. During the routing of the wires, the value of the congestion cell is incremented if the wire passes through the cell. A higher value means more wires passes through that cell. The value of the most congested cell (the most congested area) is then chosen as the final metric, **max_congestion**.

The reliability model is based on Black's equation [16]:

$$MTTF = \frac{A}{J^2} e^{\frac{E_a}{KT}} \tag{3.5}$$

where T is the metal temperature in Kelvin and J is the current density. A, E_a and K are constants with A being the cross-sectional area of the interconnect, E_a being the activation energy (e.g. 0.5eV), and K being the Boltzmann constant ($8.62 \cdot 10^{-5} eV/K$). MTTF is the mean time to failure of the wire due to electromigration.

To capture the exponential relationship between the wire reliability and temperature we take the relative reliability compared to a wire at room temperature and as shown in [7] using Equation (3.5) we get:

$$MTTF_r(w) = \frac{MTTF_{room}}{MTTF_{wire}} = exp\left[\frac{E_a}{K}\left(\frac{1}{T_{room}} - \frac{1}{T_{wire}}\right)\right]$$

where T_{wire} is the temperature of the wire under subject, w. To digest the array of relative reliabilities to one number we take the average, and to avoid having small wires leave a huge impact we multiply the relative reliability for all wires with the length of the wire giving us a final reliability metric:

$$\textit{Reliability cost metric} = \frac{1}{N} \sum_{w \in C} MTTF_r(w) \cdot l_w$$

C is the entire circuit, l_w is the length of the wire and N is the total number of wires. Notice that in this respect, it is considered a new wire every time a path crosses into another block, meaning that a path going from A to B through 4 blocks, will be considered 4 wires.

Unlike CF1 and CF2 we now have five weighting parameters, λ_A, λ_T, λ_W, λ_C, and λ_R, giving us the final cost function shown below in Equation (3.6):

$$\begin{aligned} CF3 = &\lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_delay + \\ &\lambda_C \cdot max\_congestion + \lambda_R \cdot reliability \end{aligned} \tag{3.6}$$

3.4 Choosing the Simulation Parameters

In Section 3.2.2 it is mentioned how the simulation relies on a number of parameters (k, c, and Δ_{avg}) to tweak the temperature scheduling of Fast-SA (Fast Simulated Annealing) and in Section 3.3 further parameters (λ_A, λ_T, λ_W, λ_C, λ_R), used to weigh the evaluation metrics, are introduced. In this section it is first described how to the determine the λ parameters and then second the scheduling parameters.

3.4.1 Choosing the lambda values

As stated in Section 3.3, six values (area, Tmax, wire_length, wire_delay, congestion, and reliability) are used to evaluate the floorplans using three different cost functions (CF1, CF2, and CF3) repeated here for ease of reading:

$$CF1 = \lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_length$$

$$CF2 = \lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_delay$$

$$CF3 = \lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_delay + \lambda_C \cdot max\_congestion + \lambda_R \cdot reliability$$

Common for all three cost functions is that the initial values are set so that we get a total cost of around 1000 and so that each of the metrics have values corresponding to our weighting of the metrics. For CF1, area is very important so we set the area metric to have the highest value ($\lambda_A \sim 550$), Tmax is the second most important ($\lambda_T \sim 300$) and finally wire_length is the least important ($\lambda_W \sim 150$). Because of the nature of simulated annealing and in particular Fast-SA (Section 3.2.2) the result found this way is not necessarily the best possible floorplan - it could have been stuck in a local optimum or it could have skipped the best solution since it's not trying every possible temperature schedule. Note also, that because of this, setting the area metric to 1000 and the rest to 0, might not give the floorplan with the smallest area. It will definitely not give a very good all round solution as it will most likely quickly find a local optimum and remain there. To alleviate this problem approximately 20 simulations are run with slightly changed λ values and the best (based on our predefined weighing) of these are chosen. A summary of the initial weightings is given in Table 3.1.

		CF1	CF2	CF3
λ_A	· area	550	500	400
λ_T	· Tmax	300	125	175
λ_W	· wire_length	150	-	-
λ_W	· wire_delay	-	375	150
λ_C	· congestion	-	-	175
λ_R	· reliability	-	-	100

Table 3.1: Summary of Initial Weighting of the 6 Metrics

Recall that in the original HotFloorplan cost function (CF1) only wire length

is used for evaluation, not wire delay, which is why λ_W is multiplied with wire_length in CF1 but wire_delay in CF2 and CF3.

3.4.2 Choosing the scheduling parameters

As stated in Section 3.2.2, three values (k, c, and Δ_{avg}) are used to determine the temperature schedule of the simulated annealing. As with the λ values we start out with a set of standard initial values and then tweak them later. In [30] it is suggested to use c = 100, k = 7 and a "high" initial temperature, T_0. Hotfloorplan uses an initial acceptance probability of $P = 0.99$ and an average cost of $\Delta_{avg} = 1$. From Equation (3.2), we see that this gives us an initial temperature of:

$$T_0 = \frac{-\Delta_{avg}}{\ln(P)} = \frac{-1}{ln(0.99)} = 99.5$$

However, during the work of this thesis, it became obvious that this didn't work with Fast-SA as the simulations only very rarely passed stage 2. Setting $\Delta_{avg} = 2$ fixed this in most simulations giving us an initial temperature of:

$$T_0 = \frac{-\Delta_{avg}}{\ln(P)} = \frac{-2}{ln(0.99)} = 199.0$$

To determine whether any of these parameters had to be changed from their initial values a test simulation was run using initial λ values every time a new benchmark or CF was being simulated. If the simulation went through without stopping in stage 2 while at the same time reaching a low enough temperature (T < 1.00) in stage 3 to be sure the local optimum was found, nothing was changed. If the simulation stopped in stage 2, either Δ_{avg} was raised to increase initial temperature or k was lowered to have fewer steps in stage 2 (remember that the temperature drops quickly in stage 2). The third parameter, c, used to control the speed of which the temperature decreases in stage 2 was kept at c = 100.

As mentioned in Section 3.4.1, approximately 20 simulations are run to find the best solution but if a simulation stops prematurely the scheduling parameters are tweaked to avoid this and the simulation is run again. Finally when the best set of λ's have been found the scheduling parameters are tweaked once again to find the very best solution. With approximately 20 base simulation runs and the extra simulations coming from tweaking the scheduling parameters as many as 50 simulations has to be run to get just one CF for only one benchmark which makes the implementation of Fast-SA so important.

Chapter 4

Models Implemented

Initially HotFloorPlan uses three simple models (total chip area (area), maximum substrate temperature (tmax), and wire length) to evaluate the "cost" of a floorplan. In this thesis, however, the evaluation has been greatly improved and three new models have been implemented: wire delay, congestion, and reliability. In this chapter, first the original three models are briefly outlined and then the three new models are fully presented and derived. The wire delay model is implemented based on the work done by [10].

4.1 Existing HotFloorplan models: Area, Tmax and Wire Length

As mentioned in Section 3.3, HotFloorplan(HFP) initially uses three metrics to evaluate the cost of the floorplans (CF1), namely **total area, maximum substrate temperature**, and **wire length**.

These metrics are based on three simple models:

Area:	Total area of the chip, width · height in M^2.
Tmax:	Maximum substrate temperature, the highest temperature

across the chip in K

Wire length: Total wire length of all wires, w, on the chip, C, $\sum_{w \in C} length(w)$

The models for Area and Tmax remain in all three cost functions while wire length is only used in the first cost function.

4.2 Wire Delay Model

Originally, in HotFloorplan the wire delay model is based only on wire length; longer wire equals higher delay. With todays IC design this becomes widely inaccurate and is no longer a feasible approximation due to large thermal gradients and higher temperatures. The model used in this thesis not only accounts for the temperature inside the wires (instead of only the maximum substrate temperature across the chip, tmax) but also electrical resistivity. The model is implemented based on the work done by [10].

According to [31], assuming that the substrate of each block has a uniform profile; the temperature within the interconnects can be written as:

$$T(x) = T_{sub} + \frac{\theta}{\lambda^2} \cdot \left(1 - \frac{\sinh \lambda x + \sinh \lambda (L - x)}{\sinh \lambda x}\right), 0 < x < L \tag{4.1}$$

$$\lambda^2 = \frac{1}{k_m} \left(\frac{k^*_{ins}}{t_m t_{ins}} - \frac{I^2_{rms} \rho_i \beta}{w^2 t^2_m}\right) \tag{4.2}$$

$$\theta = \frac{I^2_{rms} \rho_i}{w^2 t^2_m k_m} \tag{4.3}$$

The constants θ and λ are dependent on the chosen metal layer in a specific technology node. k_m and k^*_{ins} are the thermal conductivities of the metal and the insulator, t_m and t_{ins}, is the thickness of the metal and the insulator respectively. w is the width of the interconnect, I_{rms} is the average current density in the interconnect and ρ_i is the electrical resistivity of metal at nominal temperature. T_{sub} is the temperature of the substrate, T(x) is the temperature of the wire at position x, and L is the total length of the wire.

When we have interconnects of lengths larger than the heat diffusion length, the peak temperature is equal to $\frac{\theta}{\lambda^2}$. Because the top metal layers are used for

global wiring, the distance from the the substrate is large (larger t_{ins} than for local wires) resulting in a higher temperature rise in global wires.

According to [24] we have the following equations, The electrical resistance of metal has a linear relationship with its temperature and can be expressed as:

$$R(x) = R_0(1 + \beta \cdot T(x)), 0 < x < L \tag{4.4}$$

where R_0 is the resistance at reference temperature, β is the temperature coefficient ($1/^oC$). T(x) is the temperature profile along the wire (Equation (4.1)). As an example, the value of β is $3.9 \cdot 10^{-3}$ for copper, meaning that for every 10^oC temperature increase, the resistance also increases by 3.9%.

Using the distributed RC Elmore delay model [32] we can write the signal propagation delay through an interconnect of length L, as:

$$D = R_d \left(C_L + \int_0^L c_0(x)\, dx \right) + \int_0^L r_0(x) \cdot \left(\int_x^L c_0(\tau) d\tau + C_L \right) dx \tag{4.5}$$

where $c_0(x)$ and $r_0(x)$ are the capacitance and resistance per unit length at local x, C_L is the load capacitance, β is the temperature coefficient, and R_0 is the driver cell's ON resistance.

It can be assumed that $c_0(x)$ and $r_0(x)$ are constant, therefore, by combining equation Equation (4.4) and (4.5) we get the final temperature dependent interconnect wire delay model, D:

$$D = D_0 + (c_0 L + C_L) r_0 \beta \int_0^L T(x)\, dx - c_0 r_0 \beta \int_0^L x \cdot T(x)\, dx \tag{4.6}$$

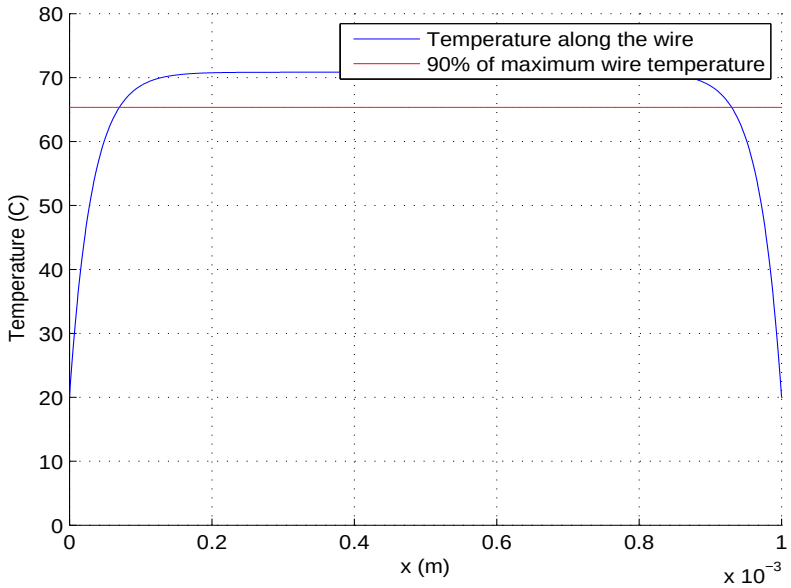
where

$$D_0 = R_0(c_0 L + C_L) + \left(c_0 r_0 \frac{L^2}{2} + r_0 L C_L \right) \tag{4.7}$$

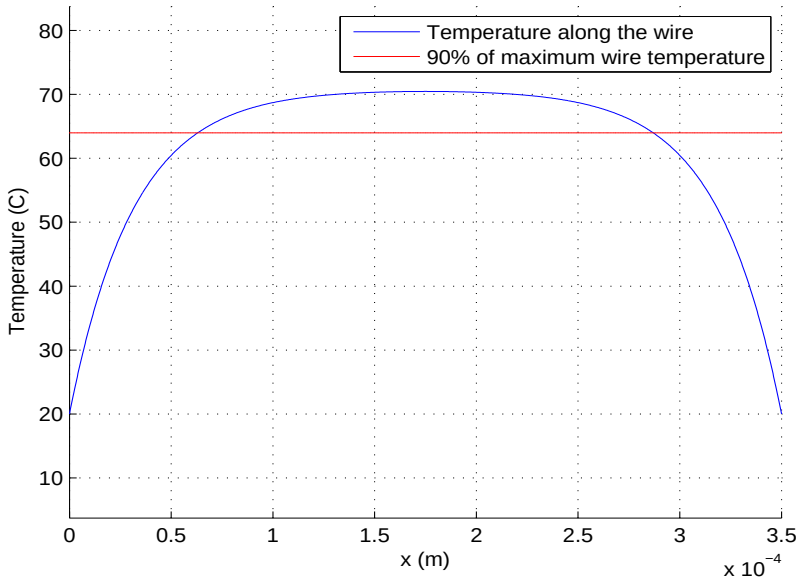
is the Elmore delay of the interconnect corresponding to the unit length resistance at 300K (reference temperature)

4.2.1 Determining the temperature, T(x)

Figure 4.1 (a) shows an example of T(x), more specifically the temperature profile along a 1000 μm global wire [24]. The wire is colder in the beginning and the end due to the fact that the contacts are closer to the substrate but after a short while the wire reaches a stable maximum temperature which it holds for a large part of the wire until it gets colder again towards the end. In



(a) 1000 μm wire with 14% of the wire under 90% of maximum temperature



(b) 350 μm wire with 36% of the wire under 90% of maximum temperature

Figure 4.1: Temperature profiles along the length of a (a) 1000 μm global wire and (b) a 350 μm global wire, resulting in 14% and 36% of the wire respectively having under 90% of the maximum wire temperature. Blue is temperature along the wire and the red line marks 90% of the maximum temperature.

this model we assume that 'maximum' temperature has been reached as soon as the temperature is at least 90% of the real maximum temperature and in this example only 14% of the wire is below the maximum temperature.

We notice in Equation (4.6) that we have to find the integral over T(x) twice. This is not only demanding to implement in C, compared to if T(x) was constant, it requires substantially more computation time. Considering that we have to calculate T(x) twice for every wire passing a block for every floorplan created, this is not trivial. With *apte* (benchmark with least number of blocks), approximately:
2 calculations · 200 wires/floorplan · 7000 floorplans/step · 500 steps/simulation = 1.4 * 10^9 calculations/simulation are needed.
Assuming T(x) to be constant would reduce runtime and complexity considerably, and since we just showed that 100%-14% = 86% is constant at 'maximum' temperature this is not a bad assumption. Consequently, with L being the total length of the wire, using T(L/2) as a constant value is a good approximation. This was, however, only in the case of a 1000 μm wire.

In Figure 4.1 (b) a 350 μm wire is shown and since this wire is shorter, only a smaller part of it is in the stable maximum temperature area, in fact 36% of the wire is below the 90% marker. With this wire length, assuming a constant T(x) does not give as good an approximation and to assure that T(x) can be assumed constant without too large an error, we have to analyse the wire lengths of the benchmarks. It should be noted that due to the nature of T(x), an increasing wire length always improves our approximation (faster rise and fall times) and decreasing it always makes it worse (slower rise and fall times). Setting a wire length of 350 μm as the minimum wire length for the approximation to hold we see that only ami33 has a noticeable amount of wires under 350 μm, more specifically $\frac{852}{5900} \cdot 100\% = 14.44\%$. The floorplan used for this analysis is found using the original HotFloorplan algorithm (CF1), as it seeks to minimize wire length. The length of each wire is defined as the Manhattan distance between the start- and end-point of the wire since the Manhattan distance is the shortest L-shaped distance between two points (recall that we use only L-shaped routing). This will create as short wires as possible and therefore serve as a worst case scenario for our approximation. In Table 4.1 are the results of our analysis.

With *apte*, *hp*, and *xerox*, the approximation holds. With *ami49*, which has the largest number of blocks, 7 wires are shorter than 350 μm required and only 18 wires shorter than 1000 μm. Ami33 has 14.44% of the wires shorter than 350 μm which is still within an acceptable limit.

		Length of wires		
Benchmark	total # of wires	>1000μm	[1000μm:350μm]	<350μm]
ami49	953	935	11	7
ami33	5900	1229	3819	852
apte	337	337	0	0
hp	553	386	167	0
xerox	3723	3651	72	0

Table 4.1: Wire length analysis of the benchmarks. The three last columns show the length of the wires.

4.3 Congestion Model

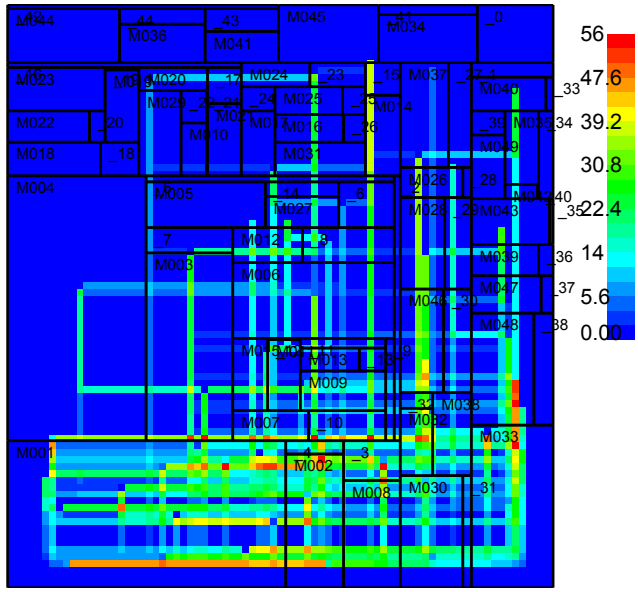
To evaluate the congestion, a simpler model has been adapted. The floorplan is divided into a $80\mu\text{m} \cdot 80\mu\text{m}$ cell grid map with every cell in the grid having an initial value of 0. Whenever a wire is created the cells that this wire passes are found and the corresponding cells have 1 added to their value. In the end we will have a congestion map with each cell having a value equal to the number of wires passing through it. The congestion map is essential since in our algorithm more wires are likely to be routed in regions with a low temperature, potentially causing routing congestion. The pseudocode for the implementation can be seen in Listing 4.1[1].

Listing 4.1: Pseudocode for the implementation of the congestion model

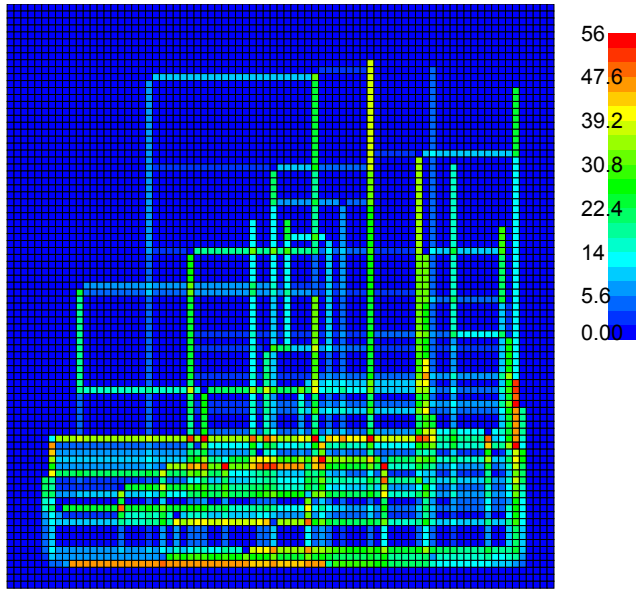
```
for every block, i in the grid
    congestion[i] = 0

for every block, i in the grid
    if wire runs through block i
        congestion[i] = congestion[i] + 1

return max_congestion
```

Two examples of the same congestion map (from ami49) is shown in Figure 4.2. In (a) the functional blocks of the benchmark are outlined and in (b) the grid map is outlined. On the map, blue means none or very little congestion and red means that we have a congestion hotspot. The maximum value in the congestion matrix (which would be one of the red blocks) is what corresponds to the evaluation metric *max_congestion* used during floorplan evaluation (Section 3.3).

[1]In the actual sourcecode - the algorithm is not implemented as one separate function but as part of other functions



(a) Functional blocks outlined



(b) Grid map outlined

Figure 4.2: Example congestion maps. The functional blocks have been outlined in (a) and the grid map has been outlined in (b)

4.4 Reliability Model

The reliability model is based on Black's equation [16]:

$$MTTF = \frac{A}{J^2} e^{\frac{E_a}{KT}} \tag{4.8}$$

where T is the metal temperature in Kelvin and J is the current density. A, E_a and K are constants with A being the cross-sectional area of the interconnect, E_a being the activation energy (e.g. 0.5eV), and K being the Boltzmann constant ($8.62 \cdot 10^{-5} eV/K$). MTTF (Mean Time To Failure) is the average time to which a wire fails due to electromigration. Because of A and J, Black's equation is technology dependent.

To capture the exponential relationship between the wire reliability and temperature, we find the relative reliability, $MTTF_r$, of the wire, w, at a reference temperature (room temperature = 300K) compared to the actual temperature. As shown in [7] using Equation (4.8), we get the following relation:

$$MTTF_r(w) = \frac{MTTF_{room}}{MTTF_{wire}} = exp\left[\frac{E_a}{K}\left(\frac{1}{T_{room}} - \frac{1}{T_{wire}}\right)\right] \tag{4.9}$$

where T_{wire} is the temperature of the wire under subject, w. The two parameters A and J cancel out, leaving only well known constants (E_a, K, and T) in the equation. To get this digested into one metric to use for evaluation, as explained in Section 3.3, we take the average ratios of all the wires. To get the final reliability metric every wire ratio is multiplied with the length of said wire to avoid small wires from having an excessive impact on the average.

This gives a cost for a wire, w, in the circuit, C:

$$MTTF_{cost}(w) = MTTF_r(w) \cdot l_w \tag{4.10}$$

and the final reliability cost metric:

$$\textit{Reliability cost metric} = \frac{1}{N} \sum_{w \in C} MTTF_{cost}(w) \tag{4.11}$$

which is the average of all wire costs in the circuit. N is the total number of wires. Recall that for the sake of simplicity, in this thesis, it is considered a new wire every time a path crosses into another block, meaning that a path going from A to B through 4 blocks, will also be considered 4 wires.

Chapter 5

Experiments

In this chapter we first describe the main points behind the actual C code implementation as well as a brief overview of the testing and verification process. After this, the results of the simulations are summarized in a table and thermal- as well as congestion- maps of each benchmark are shown. The results are first, looking at Table 5.2, briefly commented and in Section 5.3 a more detailed analysis of each individual benchmark is given. Finally, the results are summarized in Section 5.4.

5.1 Work done in C

As mentioned earlier, the models and algorithms were implemented using C coding. All changes were made to *flp.c* directly into the source code of HotFloorplan. The three models for wire delay, congestion, and reliability as well as several functions used to output statistics and general information about the simulation results have been implemented. Around 2000 lines of code has been added making it impossible to describe everything in the thesis or to add the code in an appendix. Instead a CD with the source code is included and the source code itself is well commented should the reader be interested in studying it.

One important function to mention, though, is *set_new_metrics*. This is the central function called to evaluate the floorplan for both wire delay, congestion and reliability and is run once for every floorplan. First, all the arrays and variables are initialized to 0 (new floorplan), then we run through the wire density matrix to see if there are any connections.

For every connection found (*wire_density[i][j]) > 0*), first a wire is created (*create_wire()*), then the congestion (*add_congestion()*), wire delay (*wire_temp += get_flpdelay()*), and finally the reliabilty (*reliabilitytemp += get_flpreliability()*) is updated for the floorplan and returned. The sourcecode for this function has been added in Appendix C.

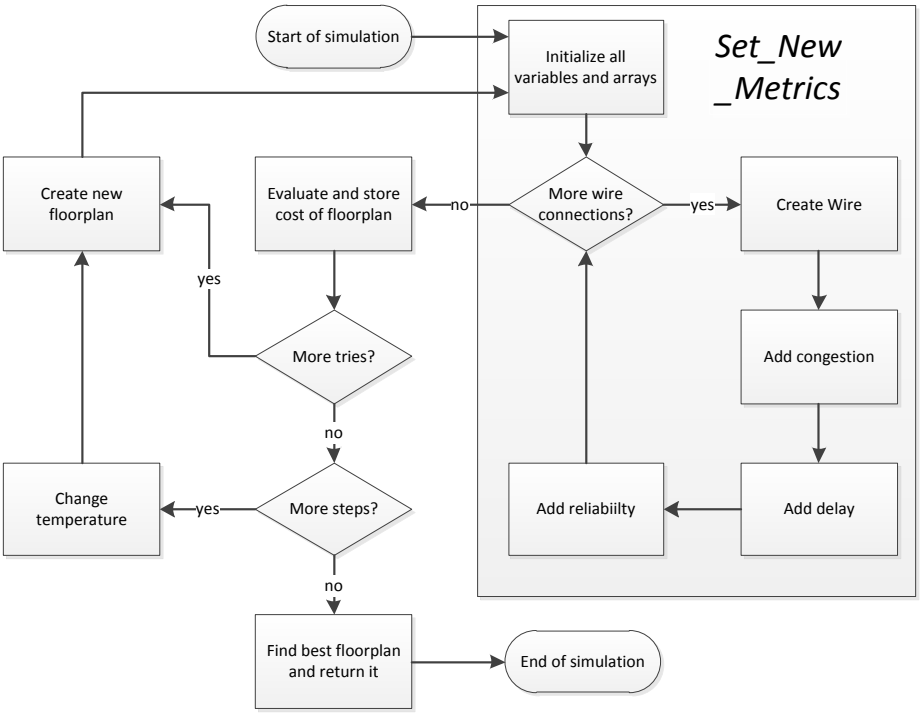


Figure 5.1: Simulation Flowchart. For evert floorplan created, all wires are found and evaluated using the function *set_new_metrics*

A simplified flowchart of the simulation is given in Figure 5.1. When the simulation starts, all variables and arrays used to evaluate the floorplan are reset and a loop going through the wire density matrix (from the benchmark description file) is initiated. For every wire connection it creates an array representing the wire and finds congestion, delay, and reliability - this is what *set_new_metrics* does.

When all connections have been evaluated, the final floorplan cost is evaluated according to the relevant cost function and the chosen λs. If more tries are left (remember we have 7000 tries/step), a new floorplan is created and the process is repeated. Otherwise, if no tries are left, we change the temperature according to the annealing schedule, create a new floorplan, and repeat the process. When all iterations of tries and steps have finished, the final best floorplan is found and outputted.

To test the validity of the source code, several functions used for testing have been implemented. Either they print out the results which can then be manually checked or they print two values, an asserted value and the actual value. If these values agree for a set of representative test cases the code is correct.

Several *perl* and *bash* scripts have been utilized too. The *bash* scripts were used to facilitate starting simulations and *perl* scripts were used to create congestion and thermal maps as well as generating power density files and power trace files for the benchmarks. The *perl* scripts were distributed with HotSpot but have been modified to serve the purpose of this thesis.

5.2 Experimental Results

As mentioned in Section 3.3, we have 3 different cost functions. Recall that:

$$CF1 = \lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_length$$

$$CF2 = \lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_delay$$

$$CF3 = \lambda_A \cdot area + \lambda_T \cdot Tmax + \lambda_W \cdot wire\_delay + \lambda_C \cdot max\_congestion + \lambda_R \cdot reliability$$

As explained in Section 3.4, for every cost function and for every benchmark a number of simulations are run to find the best set of weighting and annealing parameters. In the end, when the simulation has been fully tuned the results are stored in a log file and an example log file, clearly showing the temperature schedule, can be found in Appendix D. All the simulations have been conducted using the MCNC benchmarks with 50nm process parameters and these are given in Appendix A.

To evaluate CF2 and CF3 they are compared against the original HotFloorplan implementation (CF1). An overview of the experiment results is given in Ta-

ble 5.2 where *Cmax* and *Ravg* in the last two columns are max_congestion and reliability in CF3 respectively. Both are normalized with respect to CF1.

Table 3.1 shows the weighted priorities of the metrics and is repeated below:

		CF1	CF2	CF3
λ_A	· area	550	500	400
λ_T	· Tmax	300	125	175
λ_W	· wire_length	150	-	-
λ_W	· wire_delay	-	375	150
λ_C	· congestion	-	0	175
λ_R	· reliability	-	0	100

Table 5.1: Weighted priority of the metrics. As an example, in CF1 the metrics are weighted: area > Tmax > wire_length

From Table 5.2 we see that using delay (CF2) instead of wire length (CF1) to measure wire performance always produces floorplans with a smaller total delay. The reduction in total wire delay can be quite significant, 10% decrease in *ami49* and 20% decrease in *xerox* to give some examples. The return cost is usually a slight increase in area and wire length. In *xerox* the area overhead is 1.02% and in *ami49* where there is considerably more blocks the area in CF2 is increased by 1.01% from CF1.

On the other hand, due to the thermal awareness in the evaluation algorithm, there is always an improvement in reliability of wires in CF2 and CF3 (except for CF2 in *ami33*). Recall that reliability is relative to room temperature and looking at Equation (4.9) we see that a smaller value indicates a longer MTTF (Mean Time To Failure).

In Section 4.3 it is mentioned how the thermal aware routing around hotspots might cause increased congestion in the colder areas. In *ami49*, the maximum congestion is increased by 15% which can potentially cause a routing problem. This is not a representative case as for the other benchmarks we only have a considerably smaller congestion increase (maximum 4%) and this problem completely disappears when congestion is also taken into consideration (CF3) with a 6.70% area overhead compared to CF2 for *ami49*.

In Section 5.3, thermal- and congestion- maps are given for the benchmarks as well as further comments.

Benchmark	*# Blocks*	*Cost Function*	*Tmax (K)*	*Area (mm^2)*	*Wire length (m)*	*Tot. Delay (μs)*	*Cmax*	*Ravg*
		CF1	386.7	39.9	5.518	5.55	1.00	1.00
ami49	49	CF2	387.4	40.3	5.34	5.05	1.15	0.84
		CF3	389.5	43.0	5.751	5.18	0.71	0.82
		CF1	394.0	1.3	8.438	6.02	1.00	1.00
ami33	33	CF2	396.4	1.3	8.172	5.80	1.01	1.01
		CF3	385.1	1.5	9.096	6.42	0.49	0.67
		CF1	378.3	48.2	2.834	4.46	1.00	1.00
apte	9	CF2	379.4	48.2	2.723	4.20	1.00	0.95
		CF3	378.7	48.4	3.146	5.07	0.72	0.95
		CF1	360.5	9.5	1.732	2.57	1.00	1.00
hp	11	CF2	358.4	10.9	2.026	2.50	1.04	0.85
		CF3	358.0	10.9	1.786	2.01	0.67	0.69
		CF1	367.9	20.9	20.315	25.03	1.00	1.00
xerox	10	CF2	368.7	21.0	20.724	19.98	0.91	0.84
		CF3	369.5	21.3	23.612	24.98	0.82	0.88

Table 5.2: Experimental Results on MCNC benchmarks using different cost functions and 50nm process parameters

5.3 Thermal and Congestion maps

In this section 5 thermal maps, one for each benchmark, along with their corresponding congestion maps are presented and commented. All maps are created using CF3 and all data about the benchmarks can be found in Table 5.2. The remaining maps for CF1 and CF2 are shown in Appendix B

5.3.1 *xerox*

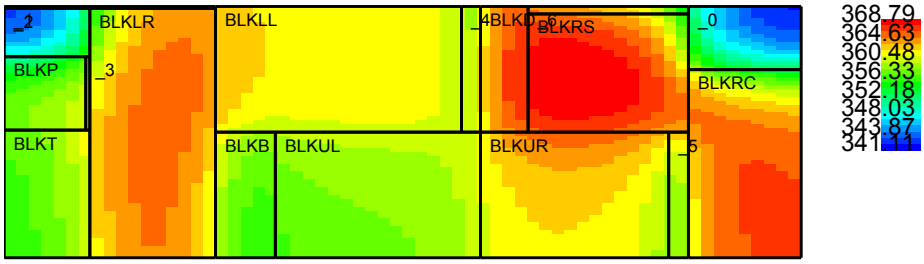


Figure 5.2: Thermal map for *xerox*

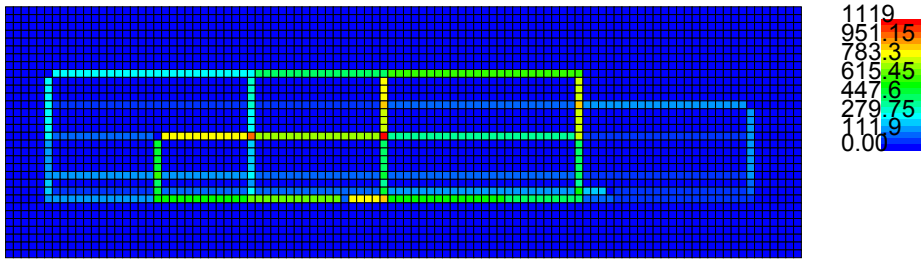


Figure 5.3: Congestion map for *xerox*

Xerox is a medium sized but very congested benchmark, in fact, it has a congestion hotspot of 1119 (1119 wires are running through one cell). This is because it has few blocks but a large amount of connections between these blocks. Add this to the fact that our routing algorithm assumes all wires are connected to the center of the blocks and the result is a high congestion.

Looking at Figure 5.3 it is easy to see that routing the wires next to each other instead of in the exact same path would result in a considerably smaller congestion. It is also worth noting that although this map does not reflect it, *xerox* has the highest total wire length - twice as much as the second highest (*ami33*), as well as the highest total wire delay - quadruple that of *ami33*.

5.3.2 *ami49*

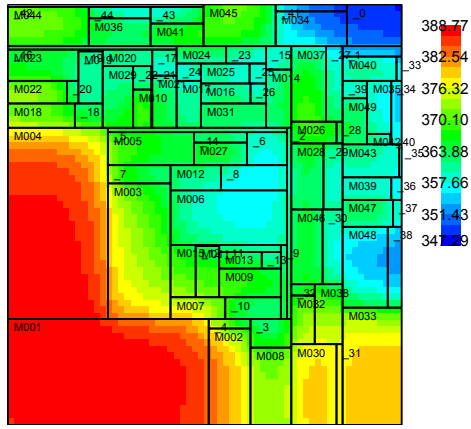


Figure 5.4: Thermal map for *ami49*

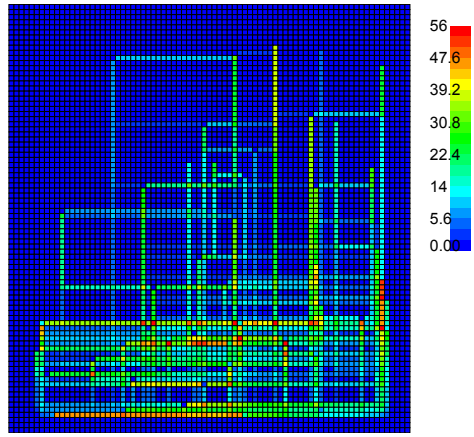


Figure 5.5: Congestion map for *ami49*

Although the area of *apte* is slightly larger, *ami49* has by far the largest number of blocks, yet the total wire length is moderate with only a few wires going between each block. This is reflected in a congestion hotspot of only 56 wires and a congestion map that *looks* congested.

Figure 5.4 shows, as expected, that the blocks holding thermal hotspots are placed in the corners of the floorplan allowing most wires to go around them.

5.3.3 *ami33*

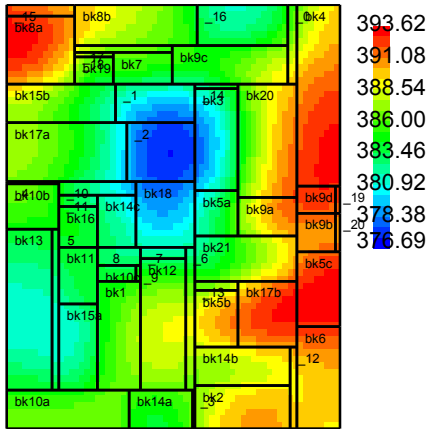


Figure 5.6: Thermal map for *ami33*

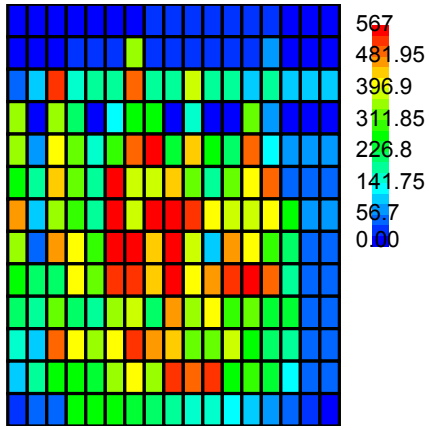


Figure 5.7: Congestion map for *ami33*

Ami33 is by far the smallest benchmark (about 1/40th that of *apte*). This is very apparent in Figure 5.7 as the map is very coarse. This is due to a, for *ami33*, large fixed cell size compared to the tiny size of *ami33*. Although it is, by far, the smallest benchmark it has 33 functional blocks (second largest after *ami49*) and also has the second largest wire length after *xerox* leaving it very congested. This reflects in a congestion hotspot of around 550 wires that is not only in one cell but over several cells.
Figure 5.6 shows that the thermal hotspots are in the sides of the floorplan in the least congested areas away from the congestion hotspot in the middle.

5.3.4 *apte*

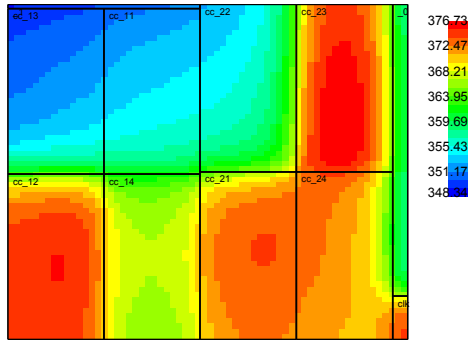


Figure 5.8: Thermal map for *apte*

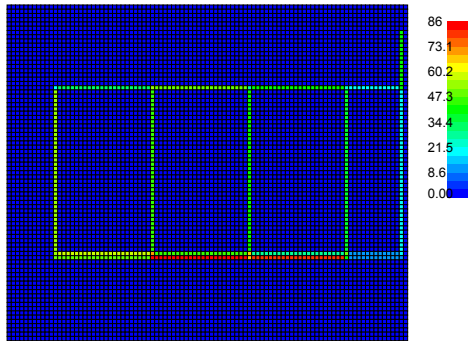


Figure 5.9: Congestion map for *apte*

This benchmark reminds of *xerox* in may ways. It has the same simple congestion map (Figure 5.9) with all wires running along the same path and the same low number of functional blocks.

It has a few key differences though. *apte* is over twice the size of *xerox* while only having around 1/8th the total wire length. This reflects in the congestion of the circuit as well as the wire delay being much lower than for *xerox*.

This benchmark further remarks itself by being the largest benchmark yet also having the least functional blocks of all. Furthermore, the blocks are very similar in shape and size. This leaves little room for our algorithm to move blocks around which shows in the moderate improvements achieved with CF2 compared to CF1.

5.3.5 *hp*

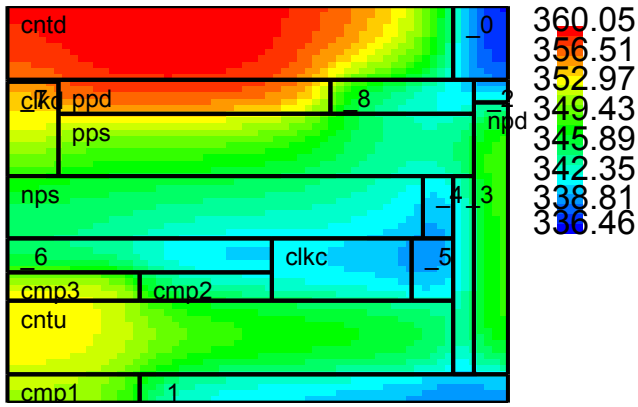


Figure 5.10: Thermal map for *ht*

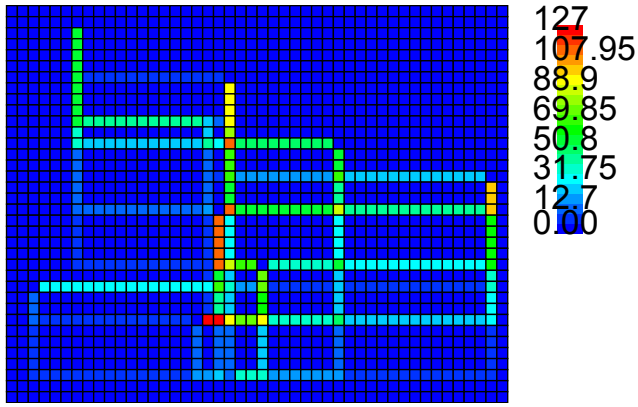


Figure 5.11: Congestion map for *ht*

This benchmark is small with little congestion and few functional blocks. Furthermore, it has the shortest wire length, the lowest peak temperature, and as a result, the lowest total wire delay. The warm block, *cntd*, is placed in the top away from the congestion hotspot in the middle.

5.4 Results Summary

A total of 15 experiments were conducted using 50nm process paramaters, one for each cost function (CF) for each benchmark. The results from each experiment are provided in Table 5.2. Furthermore, thermal- and congestion- maps have been provided for CF3 for each benchmark. These are presented in Section 5.3.

The results show that a significant decrease in wire delay can be achieved while maintaining a minimal area overhead. For *xerox* and CF2 a 20% wire delay decrease is achieved at the cost of a 0.4% increase in area. Furthermore, the congestion as well as reliability has improved.

For *hp*, on the other hand, only a 3% decrease is achieved at the cost of added congestion and an area overhead of 15%. However, looking at CF3, we have the same area overhead but a 22% decrease in wire delay as well as a 33% decrease in congestion and a 31% improvement in reliability.

With *ami33* and CF2 we see a modest 4% decrease in wire delay, however, area, congestion, and reliability remains the same. For CF3 both the wire delay and area increases which is undesirable - it is however worth noticing that congestion decreases substantially with over 50% and reliability improves with 33%

While the remaining 4 benchmarks show improvements in congestion and reliability with both CF2 (except *ami33* and *hp*) and CF3, *ami49* has a noticeable 15% increase in congestion. Although the nature of this benchmark is not very congested with only around 5.3m wire to a $40mm^2$ area, this could potentially lead to a problem with routability. Including congestion in the evaluation (CF3) removes this problem but results in a moderate decrease in wire delay of 7% at the cost of a 8% area overhead.

Looking at the maps we see that the blocks producing thermal hotspots are generally placed at the sides and corners of the die away from the congestion hotspots. It is also clear from the congestion maps that the routing is only estimated and although this leaves a somewhat skewed impression for *xerox* and *apte* the results are still useful.

Chapter 6

Design Choices and Future Work

It is always a question of relevance when choosing what to compare and how to model the effect that something has on a system. In this chapter some of the design choices and other choices made throughout this thesis are discussed.

6.1 Choosing the Models and Cost Functions

We chose to model wire delay, as the original problem was to implement the wire delay model already proposed in [24]. Models for congestion and reliability are also implemented as this is closely related to wire delay and without considering for example congestion (routability), the work becomes much less applicable. As mentioned in Section 6.5, the congestion and reliability models are effective yet simple which is because implementing more advanced models would increase runtime unnecessarily. Based on these considerations, CF1 was chosen as this was the original HFP cost function, CF2 was chosen to see the effects of the improved wire delay model, and CF3 was implemented to see the effect of the closely related metrics, congestion and reliability.

As described in Section 4.2, the final wire delay metric is the sum of wire delays

for all wires and as the number of routes does not depend on the CFs a shorter total delay also results in a shorter average delay. We assume that a shorter average delay reflects in shorter delay of the critical path and therefore a better performance. However, as the design is just floor-planned and the routing is only estimated, a full layout is needed to verify that the actual critical path is really reduced. For future work this issue could be addressed.

6.2 The Congestion Grid Map

As mentioned earlier, Hotspot is used to create a 64 x 64 cell thermal map based on the floorplan outputted by HotFloorplan. When creating the images used to display these thermal maps a script is used. This script inputs a 64 x 64 thermal map (text file) and a floorplan and outputs the thermal map image.

Initially, when creating the congestion maps, to be able to use the same script, the congestion map was fixed on a 64 x 64 cell grid map (variable size cells). However, after having run a first set of simulations it became clear that this was not a good approach as comparing congestion between the benchmarks became impossible since the benchmarks varied considerably in total chip area, consequently making the size of the cells vary considerably as well.

To remove this problem a fixed sized grid was adapted, that is, having the cells always have the same size independently of the die size to avoid having a mismatch when comparing circuits of different size. The main problem when choosing the fixed cell size is to find a balance between accuracy versus simulation runtime. As runtime increases almost linearly with number of cells and exponentially with cell size, keeping a low cell count (large fixed width) is very desirable. Finally a fixed cell size of $80\mu\text{m} \cdot 80\mu\text{m}$ was found to give the right balance.

Looking at the congestion map for *ami33* one might argue that the granularity is too coarse but it was either that or having too many cells in the large benchmarks (*ami49* and *apte*).

For future work it might be possible to adopt a more dynamic sizing that still allowed for comparison between the benchmarks. One possible way is to have a small base unit cell of maybe $40\mu\text{m} \cdot 40\mu\text{m}$ (small enough for *ami33*). This unit cell size would be used for the smallest circuits while larger circuits had a macro cell size like $80\mu\text{m} \cdot 80\mu\text{m}$ that would be easy to divide with the unit cell size. In this case 1 macro cell = 4 unit cells as shown in Figure 6.1.

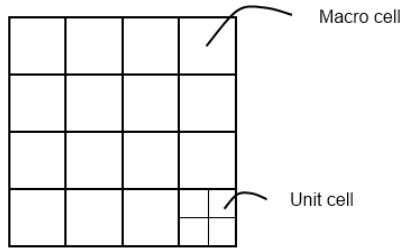


Figure 6.1: Example floorplan: 16 by 16 unit cells = 4 by 4 macro cells. For small circuits congestion would be calculated for all unit cells while for large circuits only one of the unit cells in each macro cell would be calculated.

When calculating the congestion map the small circuits would calculate the congestion of each cell as the current model does but the larger circuits, however, would only calculate the congestion for one of the unit cells of each macro block and give the remaining three unit cells the same value. That way only 1/4th of the calculations are needed and because the unit cells are so small compared to the total area of the larger chips it would not be noticeable on the printed congestion map. The smaller circuits, on the other hand, would look much better.

6.3 Classic Simulated Annealing or Fast-Simulated Annealing

Slow simulations are also what encouraged the implementation of Fast-Simulated Annealing (Fast-SA). With *ami49*, simulations took about 8-9 hours with classic simulated annealing and as described in Section 3.4 up to 50 simulations had to be run for each benchmark. Furthermore the simulations had to be refined several times as the cost functions had changed, an example of this could be when the congestion map was changed from a fixed number of cells to a fixed cell size as described in Section 6.2. This is another reason to keep the overall cell count low at the cost of a coarser congestion map for *ami33*.

On the other hand, although Fast-SA is considerably faster there is the matter of it resulting in worse floorplans. When making the transition from Classic SA to Fast-SA, several runs were made with both annealing schedules to compare them and it was found that the results produced by Fast-SA were only slightly worse. The results were also reviewed against other articles using the same

benchmarks to see if they had similar results - which they did. Furthermore, as the results are compared relative to each other it becomes less crucial to find the best absolute results. This resulted in Fast-SA being adapted.

6.4 Assuming constant T(x) with ami33

T(x) is the temperature along the length of the wire and in Section 4.2.1 it is analyzed if T(x) can be assumed to be constant for all benchmarks. *Ami33* is the only benchmark were it could be argued that the approximation is not good enough as it has a considerable amount of wires around 350 μm. It was still concluded that the approximation holds.

For future work, the implementation could be updated to also include T(x) as a function although a quick estimate suggests that runtime would be considerably increased. It would, however, serve as a great value in the analysis of the actual effects of assuming T(x) to be constant.

6.5 L-shape Routing

As mentioned in Section 2.4, we use only upper bend L-shape routing with signals always propagating from left to right leaving only one possible route for each path between two endpoints. This gives a purposely fast and simple algorithm. Implementing a more advanced algorithm would result in a slightly better routing consequently giving slightly better results but the downsides are quite compelling.

The main aim of this thesis is to estimate the effects of thermal aware wire delay, and routing (as well as congestion and reliability models) is implemented for comparison reasons. Besides taking away focus from the main content, implementing a more advanced routing model is outside the scope of this thesis. Even if a more advanced routing algorithm *was* adapted it would increase simulation runtime considerably and the impact would be reduced since the results are compared relative to each other.

For future work, however, it might be interesting to implement a more advanced routing algorithm. In [3] Maze routing is used, while in [6] they first use L-shape routing for a number of steps and then finally they use Maze routing to get the last paths routed. A third method is used in [3], which is based on setting up a

diamond around each start- and end-point that would give the same path delay as routing to the point itself using only L-shape routing. Then, according to their algorithm, the diamond is shrunk and a final point is found.

Chapter 7

Conclusion

In this project a thermal aware wire delay estimation method at the floorplanning stage which takes into account the performance degradation in wires due to thermal effects is implemented.

In nanometer technologies more and more circuits are limited by wire delay rather than gate delay, therefore optimization of wire delay is very important and sometimes crucial. Traditional thermal aware floorplanning algorithms use wire length as the evaluation metric to estimate delay and routability which has become increasingly insufficient.

The study has mainly been concerned with theory regarding thermal aware floorplanning. This involved a brief introduction to floorplanning as well as routing. We explained how MCNC benchmarks were used to evaluate the floorplans produced by HotFloorplan (HFP) and how L-shape routing was used to estimate the routability.

Then, HotFloorplan, which is a thermal aware floorplanning tool was thoroughly introduced and the underlying methods and algorithms behind it have been clarified. We explained in detail how HFP evaluates a floorplan using a cost function and how the simulation is run step-by-step using simulated annealing (SA). It was stated that classic SA results in simulations of 8-9 hours and to alleviate this problem a faster annealing schedule was implemented called Fast-

Simulated Annealing (Fast-SA). This reduced the maximum simulation runtime to about 1 hour. It was examined if the results provided by Fast-SA were adequate compared to classic SA. Fast-SA relied on a number of parameters needed to tune the simulation and it was found that careful tuning of these parameters lead to almost similar results.

To be able to evaluate the effects of temperature on wire delay, a wire delay model proposed by [10] that takes into account the thermal effects on wire delay was adapted. Models for congestion and reliability have been developed as well as these are closely related to routing and temperature. The congestion model is based on dividing the die into a grid map with each cell holding information about the number of wires passing through while the reliability model is based on the MTTF (Mean Time To failure) given by Black's equation [16]. These models have been clearly derived and presented.

To evaluate the work performed in this thesis, the three new models (wire delay, congestion, and reliability) have been implemented directly into HFP using C coding. Furthermore, the evaluation metric used in HFP has been modified to include the new models using three different cost functions. The first cost function (CF1) is the original HFP evaluation metric using only the initial models (area, maximum temperature, and wire length), cost function 2 (CF2) includes the improved wire delay model, while cost function 3 (CF3) has congestion and reliability added to its cost.

The cost functions were evaluated using the MCNC benchmarks and the results show that in the presence of thermal gradients shorter wire length does not always produce shorter delay. The results further show that a better total delay and wire reliability can be achieved with the return cost of an increase in area and wire length. Particularly *xerox* shows very promising results with a 20% wire delay decrease and only a 1.01% area overhead.

Congestion- and thermal- maps were produced for CF3 and looking at the maps it can be seen that the blocks producing thermal hotspots are generally placed at the sides and corners of the die away from the congestion hotspots. This is intended and serves as visually descriptive examples as well as confirmation that our method is indeed thermal aware. Furthermore it gives an idea of the magnitude of the problem addressed in this thesis presents by illustrating how large the variations in temperature can be. The congestion maps give a visual impression of the congestion of the benchmarks and serve as confirmation that our implementation was correct.

The work of this thesis can be used with advantage in designs where the area constraints can be loosened in pursuit of better performance. Even if area constraints are very strict, this method sees potential use. For *ami33* and *apte*

there is no area overhead yet the wire delay decreases with 9.6% and 9.5% respectively. To completely determine whether this method can be successfully applied in a specific case, a full layout including analysis of the critical path as well as routability has to be performed.

Finally, this work can serve as a stepping-stone towards a better understanding and application of thermal aware delay modelling at the floorplanning stage.

Appendix A

Constants used in the simulations

The same constants were used in all the simulations for both the wire delay model and the reliability model. The constants used in the wire delay model are given below:

Width of the wire:	$w = 0.1 \cdot 10^{-6} m$
The driver cell's On resistance:	$R0 = R_d = 10\Omega$
Load capacitance:	$C_L = 1000 fF$
Sheet capacitance at room temperature:	$c_{sh} = 0.2 fF/sq$
Sheet resistance at room temperature:	$r_{sh} = 0.077\Omega/sq$
Capacitance:	$c_0 = c_{sh}/width$
Resistance:	$r_0 = r_{sh}/width$
Temperature Coefficient:	$\beta = 3.9 \cdot 10^{-3}$
Thermal conductivity, metal:	$k_m = 386.0 W/(m \cdot K)$
Thermal conductivity, insulator:	$k_{ins*} = 0.07 W/(m \cdot K)$
Thickness, metal:	$t_m = 0.3 \cdot 10^{-6} m$

Thickness, insulator:	$t_{ins} = 0.5 \cdot 10^{-6} m$
Average current density of interconnect:	$I_{rms} = 0.9 \cdot 10^{-3} A/m^2$
Electrical resistivity at nominal temperature:	$\rho_i = 2.2 \cdot 10^{-8} \Omega m$

and for the reliability model:

Activation Energy:	$Q = 0.5eV$
Boltzmann's Constant:	$K_B = eV \cdot K^{-1}$

Appendix B

Thermal- and Congestion-maps for CF1 and CF2

In the following sections, the remaining maps are given for CF1 and CF2 as reference.

B.1 Thermal- and Congestion- Maps for *ami33*

B.1.1 Cost Function 1

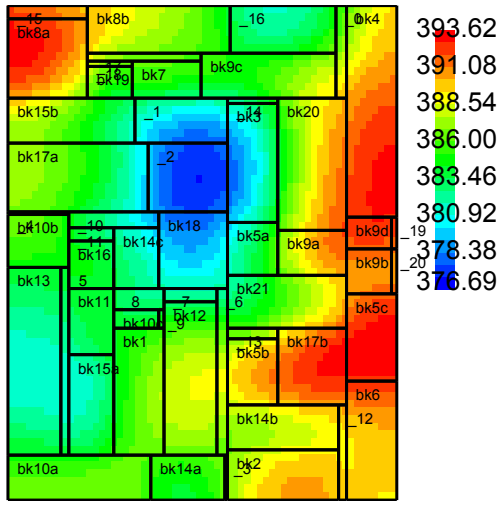


Figure B.1: Thermal map for CF1 for *ami33*

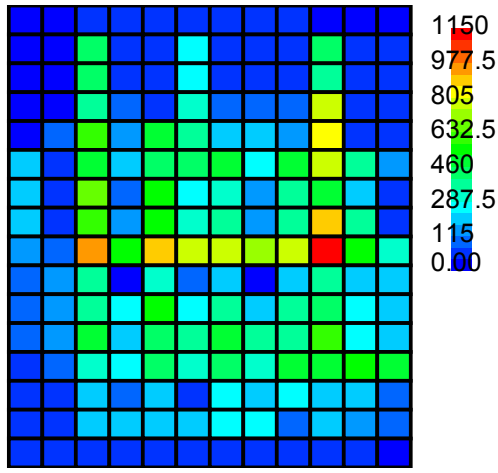


Figure B.2: Congestion map for CF1 for *ami33*

B.1.2 Cost Function 2

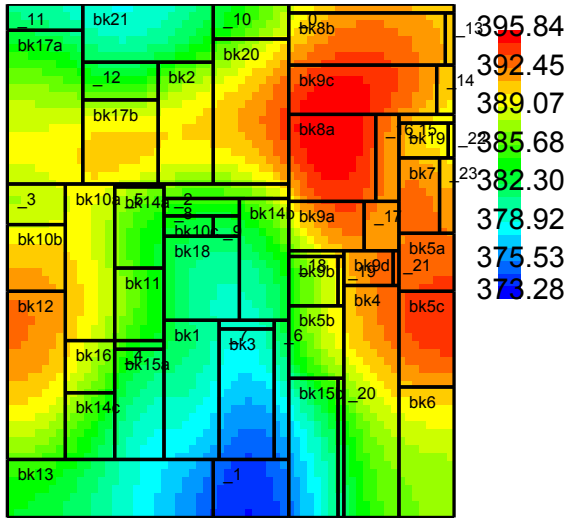


Figure B.3: Thermal map for CF2 for *ami33*

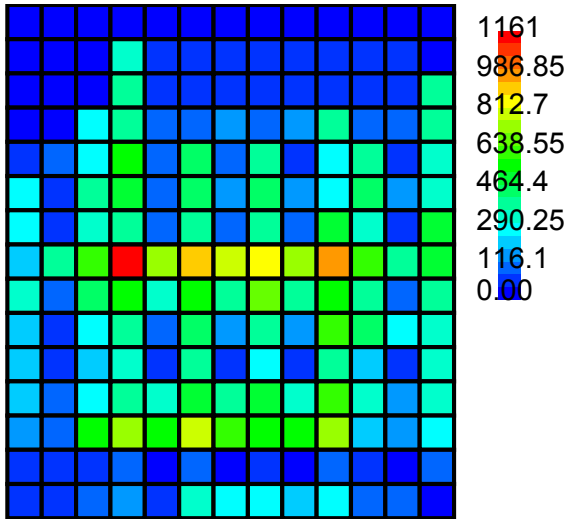


Figure B.4: Congestion map for CF2 for *ami33*

B.2 Thermal- and Congestion- Maps for *apte*

B.2.1 Cost Function 1

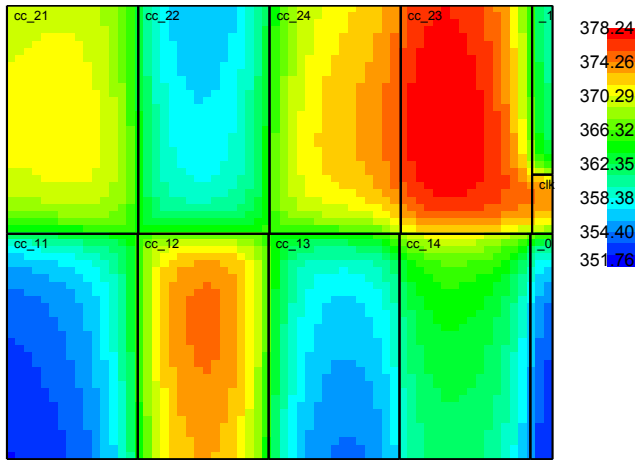


Figure B.5: Thermal map for CF1 for *apte*

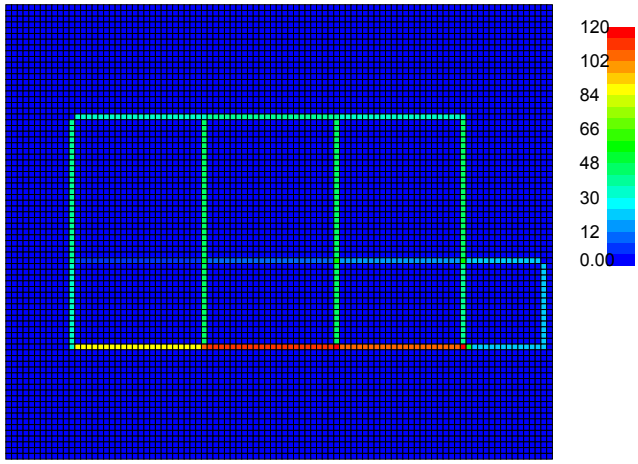


Figure B.6: Congestion map for CF1 for *apte*

B.2.2 Cost Function 2

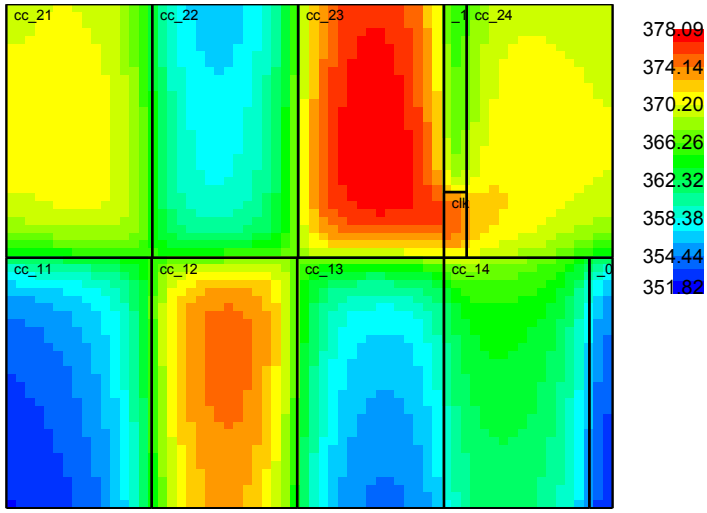


Figure B.7: Thermal map for CF2 for *apte*

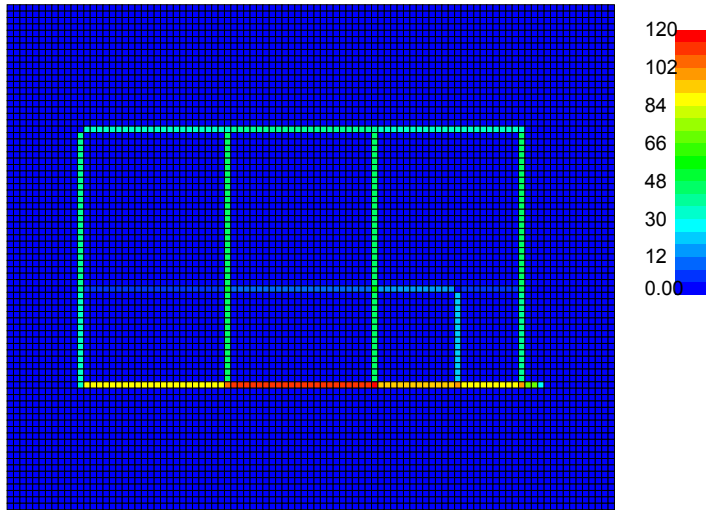


Figure B.8: Congestion map for CF2 for *apte*

B.3 Thermal- and Congestion- Maps for *hp*

B.3.1 Cost Function 1

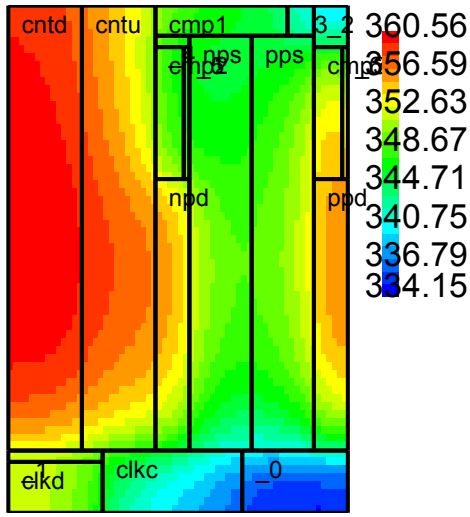


Figure B.9: Thermal map for CF1 for *hp*

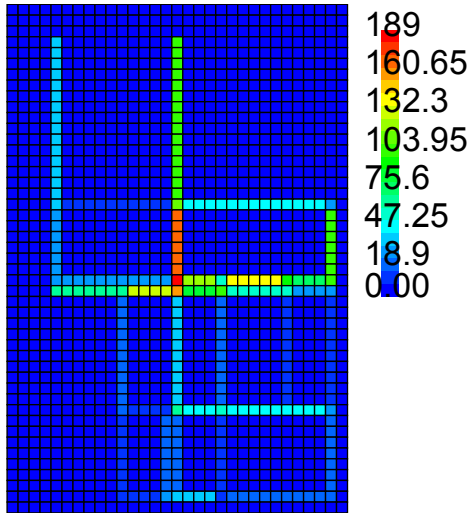


Figure B.10: Congestion map for CF1 for *hp*

B.3.2 Cost Function 2

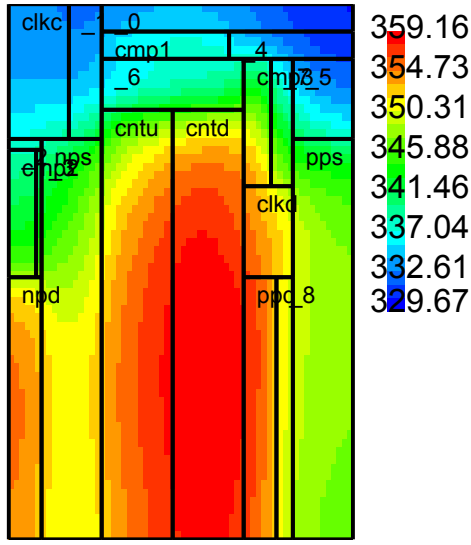


Figure B.11: Thermal map for CF2 for *hp*

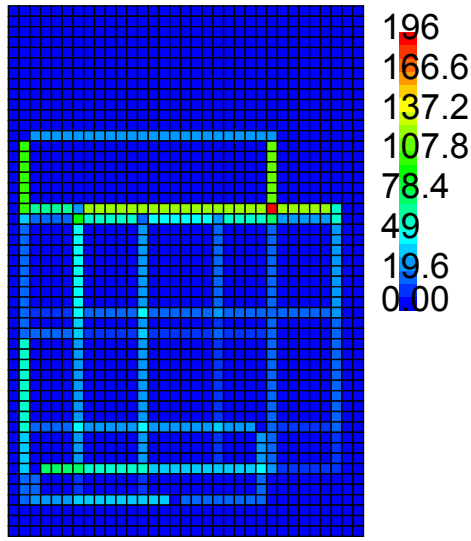


Figure B.12: Congestion map for CF2 for *hp*

B.4 Thermal- and Congestion- Maps for *xerox*

B.4.1 Cost Function 1

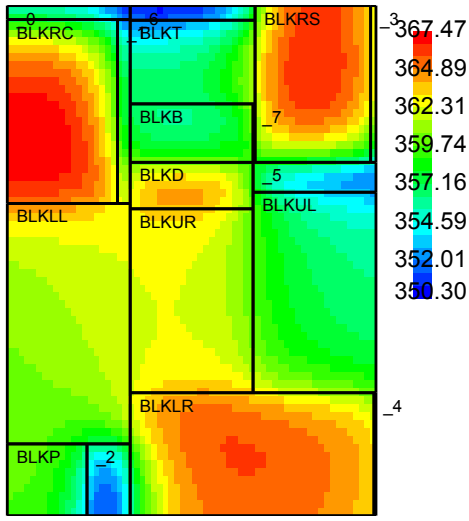


Figure B.13: Thermal map for CF1 for *xerox*

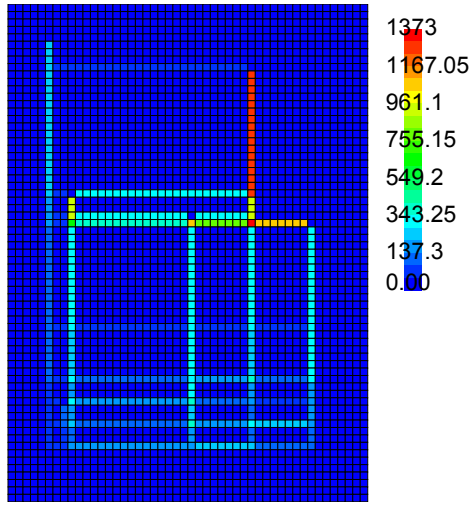


Figure B.14: Congestion map for CF1 for *xerox*

B.4.2 Cost Function 2

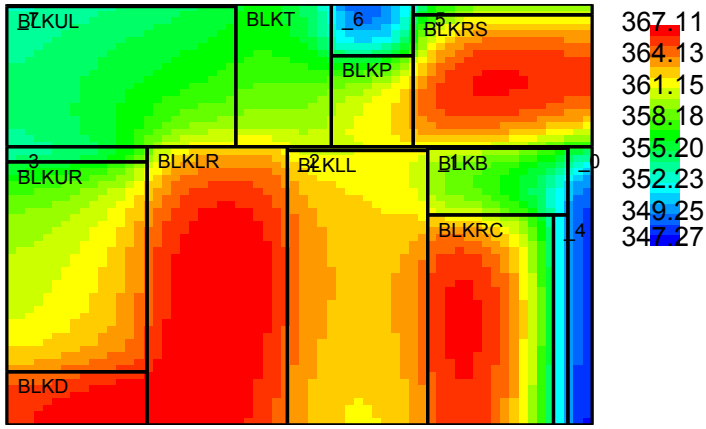


Figure B.15: Thermal map for CF2 for *xerox*

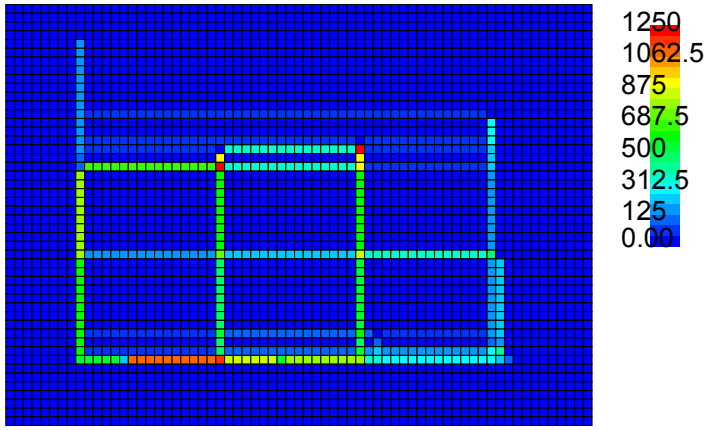


Figure B.16: Congestion map for CF2 for *xerox*

B.5 Thermal- and Congestion- Maps for *ami49*

B.5.1 Cost Function 1

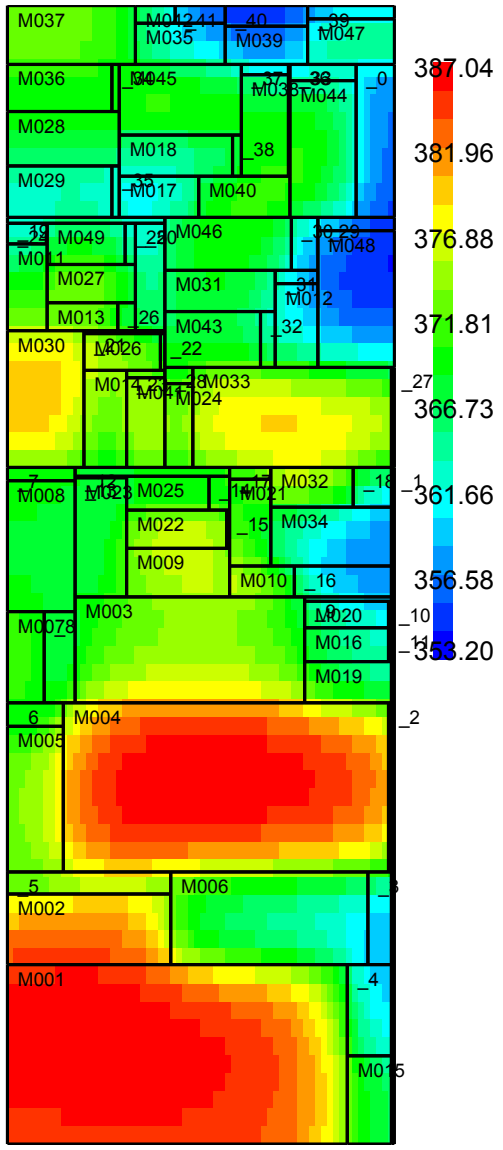


Figure B.17: Thermal map for CF1 for *ami49*

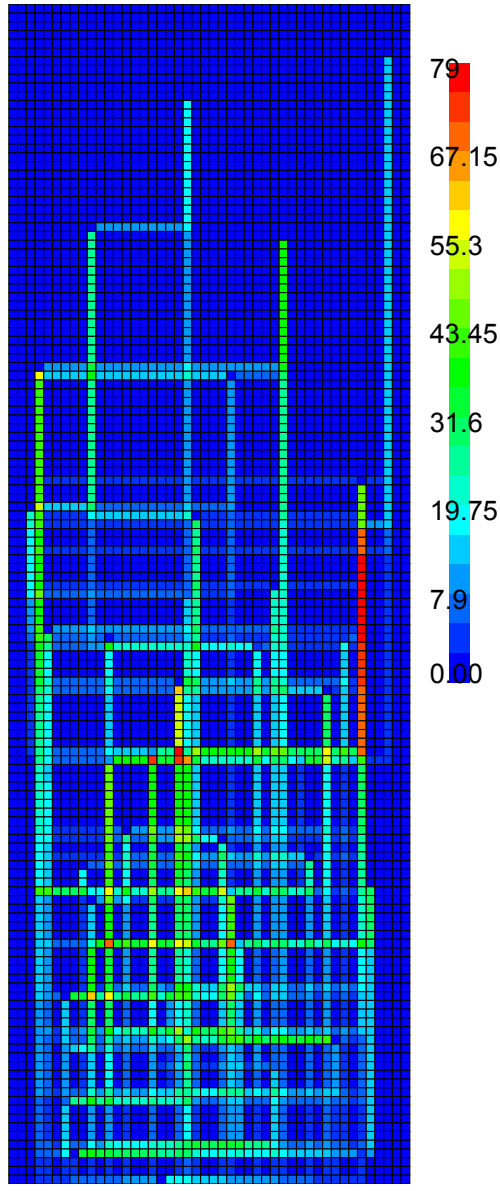


Figure B.18: Congestion map for CF1 for *ami49*

B.5.2 Cost Function 2

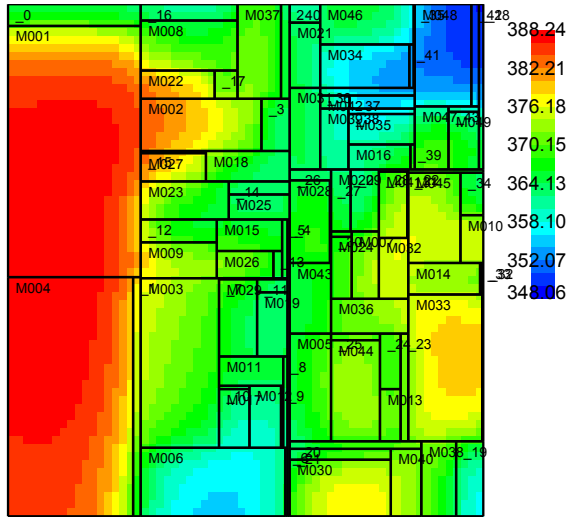


Figure B.19: Thermal map for CF2 for *ami49*

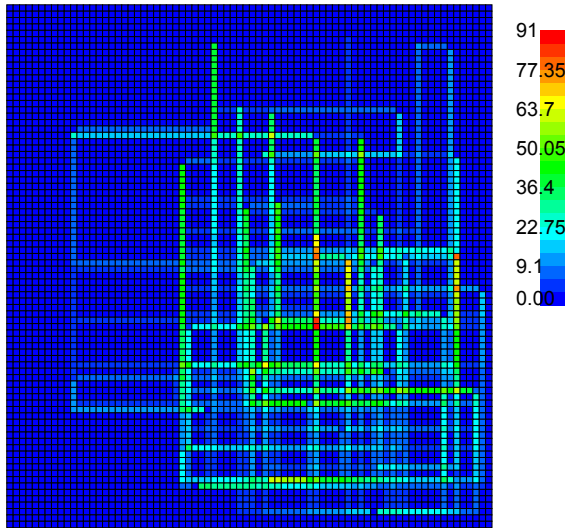


Figure B.20: Congestion map for CF2 for *ami49*

Appendix C

Source code for set_new_metrics

As mentioned in 5.1 this appendix shows the central function in the source code, *set_new_metrics*.

```
/*
 * This function sets the new metrics (evaluates) for wire delay,
 * congestion and reliability and is run once for every floorplan.
 */
void set_new_metrics(flp_t *flp, double *temp, double *↩
    block_wire_congestion, double *wire_delay, double *reliability) {

int i = 0;
int j = 0;
int k = 0;

double max_temp = get_max_blocktemp(flp, temp);
double ave_temp = get_average_blocktemp(flp, temp);
int counter = 0; //for average calculation
double tempreliability = 0.0;

wire_delay[0] = 0.0;
reliability[0] = 0.0;

for (i=0; i < 10000; i++)
  block_wire_congestion[i] = 0;

for (i=0; i < flp->n_units; i++)
  for (j=i; j < flp->n_units; j++)
    if (flp->wire_density[i][j]) { //we have a wire connection
      double *wire;
      wire=malloc(sizeof(double)*6);
      double wiretemp = 0.0;
      double reliabilitytemp = 0.0 ;
      int numberofwires = flp->wire_density[i][j];

      counter++;

      // wire[0]-[5] = startx, starty, endx, endy, cornerx, cornery
      create_wire(flp, i, j,wire);

      //add this wire to the congestion map
      add_congestion(flp, wire, block_wire_congestion, numberofwires);

      //add the delay of this wire to the total delay
      wiretemp += get_flpdelay(flp, wire, temp, i , j);

      //add this wire to the total relative reliability
      reliabilitytemp += get_flpreliability(flp, wire, temp, i, j);

      for(k=0; k < flp->wire_density[i][j]; k++) {
      wire_delay[0] += wiretemp;
      tempreliability += reliabilitytemp;

    }

  free(wire);

  }

//take the average relative reliability
reliability[0] = tempreliability / counter;

}
```

Appendix D

Example simulation log file

Here is given an example log file for ami49 using CF3. This log has k = 7 and the temperature schedule of Fast-SA is clearly visible. At stage 1 (step 0) the temperature T is very high (596.995) and this is stage 1. At stage 2 (step 1-7) the temperature falls quickly to 0.85285. Finally in stage 3 (step 8-49) the temperature starts by going back up to 7.46244 and then it slowly decreases until the algorithm ends at T =1.21836.

```
running ami49 simulation!
k: 7, c: 100, Davg: 6
initial cost: 1294.65   initial T: 596.995      final T: 1.13243
step: 0 T: 596.995      tries: 1358     accepts: 1301   rejects: 57 ↩
       avg. cost: 1980.67      best cost: 1280.62
step: 1 T: 5.96995      tries: 1358     accepts: 383    rejects: 975 ↩
       avg. cost: 1182.79      best cost: 1111.72
step: 2 T: 2.98497      tries: 1358     accepts: 223    rejects: 1135 ↩
      avg. cost: 1112.75      best cost: 1058.43
step: 3 T: 1.98998      tries: 1358     accepts: 173    rejects: 1185 ↩
      avg. cost: 1065.86      best cost: 1057.12
step: 4 T: 1.49249      tries: 1358     accepts: 145    rejects: 1213 ↩
      avg. cost: 1068.72      best cost: 1057.12
step: 5 T: 1.19399      tries: 1358     accepts: 118    rejects: 1240 ↩
      avg. cost: 1071.04      best cost: 1057.12
step: 6 T: 0.994992     tries: 1358     accepts: 115    rejects: 1243 ↩
      avg. cost: 1055.1       best cost: 1045.02
step: 7 T: 0.85285      tries: 1358     accepts: 109    rejects: 1249 ↩
      avg. cost: 1047.93      best cost: 1043.21
step: 8 T: 7.46244      tries: 1358     accepts: 369    rejects: 989 ↩
       avg. cost: 1123.52      best cost: 1043.06
```

```
step: 9 T: 6.63328       tries: 1358     accepts: 331    rejects: 1027 ↩
      avg. cost: 1106.26       best cost: 1043.06
step: 10         T: 5.96995       tries: 1358     accepts: 360    ↩
    rejects: 998    avg. cost: 1050.86       best cost: 1023.22
step: 11         T: 5.42723       tries: 1358     accepts: 349    ↩
    rejects: 1009    avg. cost: 1064.02       best cost: 1023.22
step: 12         T: 4.97496       tries: 1358     accepts: 423    ↩
    rejects: 935    avg. cost: 1069.42       best cost: 1023.22
step: 13         T: 4.59227       tries: 1358     accepts: 337    ↩
    rejects: 1021    avg. cost: 1071.17       best cost: 1023.22
step: 14         T: 4.26425       tries: 1358     accepts: 318    ↩
    rejects: 1040    avg. cost: 1050.04       best cost: 1023.22
step: 15         T: 3.97997       tries: 1358     accepts: 279    ↩
    rejects: 1079    avg. cost: 1023.62       best cost: 1005.05
step: 16         T: 3.73122       tries: 1358     accepts: 320    ↩
    rejects: 1038    avg. cost: 1043.29       best cost: 1005.05
step: 17         T: 3.51174       tries: 1358     accepts: 256    ↩
    rejects: 1102    avg. cost: 1017.9       best cost: 1005.05
step: 18         T: 3.31664       tries: 1358     accepts: 306    ↩
    rejects: 1052    avg. cost: 1015.51       best cost: 1004
step: 19         T: 3.14208       tries: 1358     accepts: 214    ↩
    rejects: 1144    avg. cost: 999.965       best cost: 991.955
step: 20         T: 2.98497       tries: 1358     accepts: 208    ↩
    rejects: 1150    avg. cost: 996.369       best cost: 987.547
step: 21         T: 2.84283       tries: 1358     accepts: 225    ↩
    rejects: 1133    avg. cost: 997.929       best cost: 987.547
step: 22         T: 2.71361       tries: 1358     accepts: 210    ↩
    rejects: 1148    avg. cost: 999.618       best cost: 987.547
step: 23  T: 2.59563  tries: 1358 accepts: 192  rejects: 1166 avg. ↩
    cost: 994.555  best cost: 987.547
step: 24  T: 2.48748  tries: 1358 accepts: 206  rejects: 1152 avg. ↩
    cost: 1001.58  best cost: 987.547
step: 25  T: 2.38798  tries: 1358 accepts: 195  rejects: 1163 avg. ↩
    cost: 1009.53  best cost: 987.547
step: 26  T: 2.29613  tries: 1358 accepts: 237  rejects: 1121 avg. ↩
    cost: 1004.46  best cost: 987.547
step: 27  T: 2.21109  tries: 1358 accepts: 173  rejects: 1185 avg. ↩
    cost: 978.409  best cost: 966.918
step: 28  T: 2.13212  tries: 1358 accepts: 180  rejects: 1178 avg. ↩
    cost: 972.376  best cost: 966.918
step: 29  T: 2.0586 tries: 1358 accepts: 210  rejects: 1148 avg. cost:↩
     977.171  best cost: 966.918
step: 30  T: 1.98998  tries: 1358 accepts: 191  rejects: 1167 avg. ↩
    cost: 977.282  best cost: 966.918
step: 31  T: 1.92579  tries: 1358 accepts: 196  rejects: 1162 avg. ↩
    cost: 981.918  best cost: 966.918
step: 32  T: 1.86561  tries: 1358 accepts: 182  rejects: 1176 avg. ↩
    cost: 977.007  best cost: 966.918
step: 33  T: 1.80908  tries: 1358 accepts: 187  rejects: 1171 avg. ↩
    cost: 973.122  best cost: 966.918
step: 34  T: 1.75587  tries: 1358 accepts: 196  rejects: 1162 avg. ↩
    cost: 980.224  best cost: 966.918
step: 35  T: 1.7057 tries: 1358 accepts: 211  rejects: 1147 avg. cost:↩
     977.139  best cost: 966.918
step: 36  T: 1.65832  tries: 1358 accepts: 164  rejects: 1194 avg. ↩
    cost: 977.793  best cost: 966.918
step: 37  T: 1.6135 tries: 1358 accepts: 198  rejects: 1160 avg. cost:↩
     978.643  best cost: 966.918
step: 38  T: 1.57104  tries: 1358 accepts: 173  rejects: 1185 avg. ↩
    cost: 981.664  best cost: 966.918
step: 39  T: 1.53076  tries: 1358 accepts: 168  rejects: 1190 avg. ↩
    cost: 978.535  best cost: 966.918
step: 40  T: 1.49249  tries: 1358 accepts: 187  rejects: 1171 avg. ↩
    cost: 977.131  best cost: 966.918
step: 41  T: 1.45609  tries: 1358 accepts: 180  rejects: 1178 avg. ↩
```

```
    cost: 977.832  best cost: 966.918
step: 42  T: 1.42142  tries: 1358 accepts: 220  rejects: 1138 avg. ↩
    cost: 980.373  best cost: 966.918
step: 43  T: 1.38836  tries: 1358 accepts: 174  rejects: 1184 avg. ↩
    cost: 980.397  best cost: 966.918
step: 44  T: 1.35681  tries: 1358 accepts: 156  rejects: 1202 avg. ↩
    cost: 982.312  best cost: 966.918
step: 45  T: 1.32666  tries: 1358 accepts: 122  rejects: 1236 avg. ↩
    cost: 980.695  best cost: 966.918
step: 46  T: 1.29782  tries: 1358 accepts: 140  rejects: 1218 avg. ↩
    cost: 985.755  best cost: 966.918
step: 47  T: 1.2702 tries: 1358 accepts: 148  rejects: 1210 avg. cost:↩
     983.002  best cost: 966.918
step: 48  T: 1.24374  tries: 1358 accepts: 154  rejects: 1204 avg. ↩
    cost: 984.523  best cost: 966.918
step: 49  T: 1.21836  tries: 1358 accepts: 161  rejects: 1197 avg. ↩
    cost: 983.701  best cost: 966.918

thermal gradient: 39.216766
-----------------------------------------------------------------
total block_wire_congestion: 34978.000000
average block_wire_congestion: 3.497800
lambdaA:         10000000.000000
lambdaT:         0.800000
lambdaW:         20000000.000000
lambdaC:         2.000000
lambdaR:         300.000000
-----------------------------------------------------------------
lambdaA * area:         429.733915
lambdaT * Tmax:         311.604521
LambdaW * wire_delay:      103.589233
LambdaW * wire_length:        115023726.642235
LambdaC * max_wire_congestion:     112.000000
LambdaR * reliability:       9.990830
Area:           0.0000429734
Tmax:           389.505651
wire_length:          5.751186
max_wire_congestion:        56.000000
average MTTF ratio (low = good):    0.033303
Width: 0.006342, Height: 0.006776, Area: 0.0000429734
Columns: 79, Rows: 84, Total: 6636
-----------------------------------------------------------------
wire delay (us):        5.179462
-----------------------------------------------------------------
```

Appendix E

Temperature Dependent Wire Delay Estimation in Floorplanning

Accepted for publishing and presented at the 29th Norchip Conference November 2011 in Lund, Sweden.

Temperature Dependent Wire Delay Estimation in Floorplanning

Andreas Thor Winther, Wei Liu*, Alberto Nannarelli and Sarma Vrudhula†

Dept. of Informatics, Technical University of Denmark, Kongens Lyngby, Denmark
*Dept. of Computer Engineering, Politecnico di Torino, Torino, Italy
†Computer Systems Engineering, Arizona State University, Tempe, USA

***Abstract*—Due to large variations in temperature in VLSI circuits and the linear relationship between metal resistance and temperature, the delay through wires of the same length can be different. Traditional thermal aware floorplanning algorithms use wirelength to estimate delay and routability. In this work, we show that using wirelength as the evaluation metric does not always produce a floorplan with the shortest delay. We propose a temperature dependent wire delay estimation method for thermal aware floorplanning algorithms, which takes into account the thermal effect on wire delay. The experiment results show that a shorter delay can be achieved using the proposed method. In addition, we also discuss the congestion and reliability issues as they are closely related to routing and temperature.**

I. Introduction

With technology scaling, the feature sizes of both CMOS devices and wires shrink and designers are able to integrate more and more functionalities into a single chip. Delay in CMOS transistors decreases as the channel length is reduced in each new process. Delay in metal wires, on the other hand, shows different behaviors. For local wires, delay decreases as the distance between the end points becomes smaller with scaling. For global wires, which has to span across the chip, delay increases due to the fact that die size does not shrink but slightly increases in each new process. In fact, delay in global wires has increased steadily with technology scaling over the years and already dominates path delays.

In addition to technology scaling, modeling of global wires is further complicated by thermal effects. Due to the high degree of integration and aggressive power management techniques (clock gating, power gating, etc.), the power consumption in different regions of the chip (e.g. the power density) can vary significantly. The spatially non uniform power consumption within the chip exhibits as thermal gradients, which are temperature differences between different regions.

The high temperature and large thermal gradient in metal layers can affect many aspects of interconnect design, including signal delay, routing congestion and reliability. The propagation delay in metal wires is severely degraded by high temperature as the electrical resistivity in metal increases linearly with temperature. The large within-die thermal gradients result in performance mismatch between wires of the same length but subject to different temperatures. Traditional physical design algorithms such as floorplanning and routing assume resistivity in interconnects is uniform and constant. Consequently, wirelength is used as a metric to estimate signal delay and congestion of interconnects. However, in designs where the substrate has nonuniform thermal profile, the traditional way of estimating wire delay can lead to large errors. This is because wire performance decreases with an increase in temperature and the delay of two wires of the same length are no longer equal.

The thermal effect is more prominent in global wires than in local wires because global wires are routed in layers that are far away from the heat sink, and because global wires are routed for long distance and may develop a large thermal gradient. In recent years, temperature variation induced clock skew in clock distribution network has received a lot of attention. In [1], [2], the authors described design time clock tree synthesis algorithms to modify merging locations against nonuniform substrate thermal profile. While in [3], optimal insertion of tunable delay buffers into clock trees is discussed to adjust at run time the delay of clock distribution paths that are more susceptible to temperature variations. Thermal aware global routing algorithms for improving reliability are also discussed in [4], [5].

On the other side, regarding global signal wires, although extensive work has been done on thermal aware floorplanning, all of them assume electrical resistivity in wires is constant and thermal gradients in the substrate has no impact on wire delay. This assumption is in general invalid and increasingly inaccurate in nanometer high performance designs where large temperature gradients already exist in the substrate.

In this paper, we study the problem of estimating the temperature dependent wire delay during the floorplanning stage. We first illustrate the impact of nonuniform thermal profile on the delay in wires. Then we propose a new way to estimate the wire delay in thermal aware floorplanning algorithms. The proposed algorithm takes the delay, instead of the wirelength, as one of the optimization goals, in this way, mitigating the excessive delay increase caused by high temperature. In addition, we also consider the impact of routing congestion and the reliability of wires, which are important metrics in evaluating floorplans in a realistic setting.

II. Thermal Aware Floorplanning

Floorplanning is the initial stage of physical implementation of VLSI circuits, which to a large extent determines the

quality of the final design. During the floorpalnning stage, the main design tasks include macro block placement, global wire planning and Power/Ground network design. Traditional floorplanning algorithms only optimizes the total area and wirelength. In recent years, as thermal issues become more prominent, the maximum temperature is also added to the cost functions in so called thermal aware floorplanning algorithms [6], [7], [8]. Since the thermal coupling between high power consumption blocks can significantly affect the temperature distribution in the whole chip, thermal aware floorplanning algorithms consider peak temperature in addition to area and wirelength in the evaluation of a floorplan. The estimation of peak temperature usually requires the use of compact thermal models that can compute the temperature profile in a very efficient way [9].

The floorplanning tool proposed in [6], HotFloorplan, is a very representative thermal aware floorplanner. The topology of the floorplan is represented in *Normalized Polish Expression* and the optimization process is implemented as a simulated annealing process. During the annealing, for every candidate floorplan, the algorithm invokes routines in HotSpot [10] to compute the temperature distribution and uses the maximum temperature as one of the metrics in the cost function. The other metrics in the cost function are total area and total wirelength. The connectivity information in HotFloorplan is stored in a two-dimensional connectivity matrix and manhattan distance is used to estimate the wirelength between two endpoints in a wire.

III. Temperature Estimation and Wire Delay Calculation

The high temperature in global wires is caused not only by self heating, but also by heat diffusion from the substrate. According to [11], the temperature within the interconnect for a given substrate temperature can be expressed as:

$$T(x) = T_{sub} + \frac{\theta}{\lambda^2}\left(1 - \frac{\sinh \lambda x + \sinh \lambda(L-x)}{\sinh \lambda L}\right) \quad (1)$$

where θ and λ are constants for a chosen metal layer in a specific technology node and depend on the thermal conductivity of metal and insulator, on their geometries, and on the electical parameters of the interconnect (current density and resistivity).

The peak temperature rise is equal to θ/λ^2 for interconnects whose lengths (L) are larger than the heat diffusion length. As the global wires are routed in top metal layers, the distance from the substrate is larger than local wires, and, as a result, the temperature rise in global wires is higher.

The electrical resistance of metal has a linear relationship with its temperature and can be expressed as:

$$R(x) = R_0(1 + \beta \cdot T(x)) \quad (2)$$

where R_0 is the resistance at reference temperature, β is the temperature coefficient ($1/^\circ$C) and $T(x)$ is the temperature profile along the length of the wire. The value of β for copper at room temperature is 3.9E-3, which means for every 10 $^\circ$C increase in temperature, the resistance would increase by 3.9%. According to the distributed RC Elmore delay model [12], signal propagation delay through the interconnect of length L can be written as:

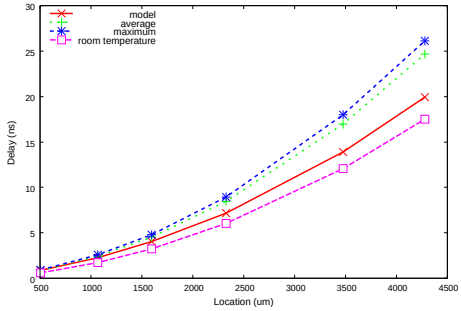


Fig. 1. Delay increase in a global wire using different thermal profiles

$$D = R_d\left(C_L + \int_0^L c_0(x)dx\right) + \int_0^L r_0(x)\cdot\left(\int_x^L c_0(\tau)d\tau + C_L\right)dx \quad (3)$$

where R_d is the driver cell's ON resistance, $c_0(x)$ and $r_0(x)$ are the capacitance and resistance per unit length at location x and C_L is the load capacitance.

By combining (2) and (3), we can obtain a temperature dependent interconnect delay model:

$$D = D_0 + (c_0L + C_L)r_0\beta\int_0^L T(x)dx - c_0r_0\beta\int_0^L x\cdot T(x)dx \quad (4)$$

where

$$D_0 = R_d(c_0L + C_L) + \left(c_0r_0\frac{L^2}{2} + r_0LC_L\right) \quad (5)$$

is the Elmore delay of the interconnect corresponding to the unit length resistance at reference temperature.

In Figure 1, we plot the delay increase in a global wire from one benchmark circuit in a 50 nm process using different thermal profiles. The red curve in solid line is the delay calculated using (4) with a thermal profile extracted from HotFloorplan. For comparison purposes, we also plot the delay increase when uniform temperature profiles are used, namely the maximum, the average and the room temperature. The delay at the end of the wire using the extracted thermal profile is 20 ns while using the maximum and average temperatures can result in errors larger than 25%. This means when estimating global wire delay, we need to take the thermal gradient and not the maximum or average temperature into consideration.

IV. Thermal Aware Wire Planning in Floorplanning

As we have discussed in Section III, the delay of global wires subject to large temperature variations is no longer linearly

proportional to wirelength. To have an accurate estimation of wire delay, the temperature effect including thermal gradients has to be considered. Since the thermal profile on the die is mainly determined by the locations of macro blocks, it is, therefore, possible to perform temperature dependent delay estimation at the floorplanning stage.

Our wire planning method is described in Algorithm 1. Given a floorplan, together with the associated connectivity matrix and the thermal map, we compute the total delay, the maximum congestion and the average reliability of all wires. The layout of each wire is determined by performing L-shape routing between the center of the connecting blocks. Once the physical layout of the wire is known, we record the blocks over which the wire is routed. The temperature profile along the wire and the wire delay are then calculated using (1) and (4). With thermal effects taken in account, the two paths in the bounding box of two end points of a wire can have different delay although their length are the same. In our algorithm, we choose the path with a shorter delay, which is different from HotFloorplan where the two paths are considered as identical. The temperature profile is also used to evaluate the wire reliability in terms of Mean Time To Failure (MTTF). In addition, a congestion map made up of a two dimensional matrix is updated with the route of the wire to evaluate the routability of the floorplan. The congestion map is useful because in our algorithm more wires are likely to be routed in regions with a low temperature, potentially causing routing congestion. We now describe the congestion map and the reliability metrics used in the algorithm.

Algorithm 1 Wire planning in floorplanning

```
INPUT: Floorplan description
INPUT: Connectivity matrix
INPUT: Thermal map
for all wires in the connectivity matrix do
    Perform L-Shape routing
    Extract the thermal profile along the wire
    Calculate wire delay using (4)
    Calculate wire reliability
    Update congestion map
end for
OUTPUT: Delay, Congestion and Reliability of Wires
```

A. Congestion Map

Our congestion map is made up of a two dimensional matrix, which divides the floorplan into a congestion grid. The size of the routing cell in the grid is 80 μm $\times$ 80 μm. During the routing of wires, the value of a cell is incremented if a wire passes through the cell. A higher value means more wires passes through that cell.

An example congestion map is shown in Figure 2, where the congestion cells are plotted in colors over the floorplan. Cells in red color indicate congestion hotspots. The maximum value in the congestion matrix can be used to evaluate the routability of the design.

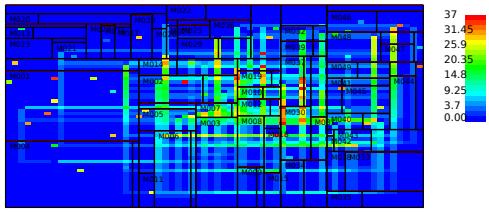


Fig. 2. Example congestion map

B. Reliability Metrics

The reliability of wires is usually measured by MTTF, which is an estimation of the average time before the wire fails due to electromigration. The MTTF can be modeled using Black's equation [13],

$$MTTF = \frac{A}{J^2} e^{\frac{Ea}{KT}} \tag{6}$$

where T is the wire temperature, J is the current density, A is related to the wire cross-sectional area and Ea and K are the activation energy and Boltzmann constants respectively.

To capture the exponential relationship between the wire reliability and temperature, we adopt the relative reliability metric used in [5], which computes the ratio of the MTTF of a wire under room temperature T_{rm} to the MTTF under temperature T_e,

$$r(e) = \frac{MTTF^{rm}}{MTTF^{e}} = exp\left[\frac{Ea}{K}\left(\frac{1}{T_{rm}} - \frac{1}{T_e}\right)\right] \tag{7}$$

In Algorithm 1, we compute the relative reliability of each wire and use the average value to evaluate the reliability of the floorplan.

V. Experimental Results

We implemented the wire planning algorithm described in the previous section in HotFloorplan and to speed up the lengthy simulation time of HotFloorplan we adopt the fast simulated annealing (FastSA) approach proposed in [14]. We used the MCNC macro block benchmark circuits with 50 nm process parameters to evaluate our proposed wire planning methods. To compare the proposed methods against the original HotFloorplan implementation, we run the floorplanning algorithm on each benchmark circuit using different cost functions.

The experiment results are summarized in Table I, where *Cmax* and *Ravg* in the last two columns are the normalized maximum congestion value and the normalized average reliability of all wires respectively. The three cost functions we used are:

- **CF1** is the original function in HotFloorplan defined as $\lambda_A \times Area + \lambda_W \times Wirelength + \lambda_T \times Tmax$
- **CF2** replaces Wirelength with WireDelay in CF1
- **CF3** adds the maximum congestion and the average reliability to CF2

In the experiment results, **CF1** and **CF2** are of the most interest. Since the weight of each metric (λ) in the cost

TABLE I
EXPERIMENTAL RESULTS ON MCNC BENCHMARKS USING DIFFERENT COST FUNCTIONS

Benchmark	*# Blocks*	*Cost Function*	*Tmax (K)*	*Area* (mm^2)	*Wirelength (m)*	*Tot. Delay* (μs)	*Cmax*	*Ravg*
ami49	49	CF1	386.7	39.9	5.518	5.55	1.00	1.00
		CF2	389.4	41.7	5.456	4.94	1.43	0.76
		CF3	389.5	43.0	5.751	5.18	0.71	0.82
ami33	33	CF1	394.0	1.3	8.438	6.02	1.00	1.00
		CF2	396.4	1.3	8.172	5.80	1.01	1.01
		CF3	385.1	1.5	9.096	6.42	0.49	0.67
apte	9	CF1	378.3	48.2	2.834	4.46	1.00	1.00
		CF2	379.4	48.2	2.723	4.20	1.00	0.95
		CF3	378.7	48.4	3.146	5.07	0.72	0.95
hp	11	CF1	360.5	9.5	1.732	2.57	1.00	1.00
		CF2	358.4	10.9	2.026	2.50	1.04	0.85
		CF3	358.0	10.9	1.786	2.01	0.67	0.69
xerox	10	CF1	367.9	20.9	20.315	25.03	1.00	1.00
		CF2	368.7	21.0	20.724	19.98	0.91	0.84
		CF3	369.5	21.3	23.612	24.98	0.82	0.88

functions can significantly alter the results, we list **CF3** only for comparison purposes. The λs we chose in **CF1** and **CF2** give order of optimization priorities from high to low as: area, peak temperature and wire performance. *Cmax* and *Ravg* are normalized to **CF1**, which give a comparison against the results obtained from the original HotFloorplan.

It can be seen in Table I that in all cases using delay (**CF2**) instead of wirelength (**CF1**) to measure the wire performance always produces floorplans with a smaller total delay. The amount of reduction in total delay can be significant, 11% decrease in *ami49* and 20% decrease in *xerox*. The cost is usually a slight increase in area and wirelength. For example in *ami49* where the number of blocks is much larger than the others, the area in **CF2** is increased by 4.6% from **CF1**.

On the other hand, due to the thermal awareness in the algorithm, there is always an improvement in reliability of wires in **CF2** and **CF3**. Note that reliability is relative to room temperature and according to (7) a smaller value indicates a longer MTTF. As mentioned in Section IV, avoiding thermal hotspots during routing might cause congestion in the cool area. From Table I we can see that, in the worst case, as in *ami49*, the maximum congestion increased by 43% can potentially cause a routing problem. When congestion is also taken into consideration (as in **CF3**), the routing can be made much easier at the cost of an increase in area and wirelength.

VI. CONCLUSIONS

In this paper, we proposed a wire delay estimation method at the floorplanning stage which takes into account the performance degradation in wires due to thermal effect. The method is implemented in HotFloorplan and evaluated using the MCNC benchmarks with congestion and wire reliability considered. The experiment results show that in the presence of thermal gradients shorter wirelength does not always produce shorter delay. The proposed method, on the other hand, can achieve a better total delay and wire reliability at the cost of an increase in area and wirelength.

REFERENCES

[1] C. Liu, J. Su, and Y. Shi, "Temperature-aware clock tree synthesis considering spatiotemporal hot spot correlations," *Proc. of 26th IEEE International Conference on Computer Design*, pp. 107–113, Oct. 2008.

[2] M. Cho, S. Ahmedtt, and D. Pan, "TACO: temperature aware clock-tree optimization," Nov. 2005, pp. 582–587.

[3] A. Chakraborty, K. Duraisami, A. Sathanur, P. Sithambaram, L. Benini, A. Macii, E. Macii, and M. Poncino, "Dynamic Thermal Clock Skew Compensation Using Tunable Delay Buffers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 6, pp. 639–649, June 2008.

[4] A. Gupta, N. Dutt, F. Kurdahi, K. Khouri, and M. Abadir, "Thermal Aware Global Routing of VLSI Chips for Enhanced Reliability," *Proc. of 9th International Symposium on Quality Electronic Design*, pp. 470–475, Mar. 2008.

[5] K. Lu and D. Pan, "Reliability-aware global routing under thermal considerations," *Proc. of 1st Asia Symposium on Quality Electronic Design*, pp. 313–318, July 2009.

[6] K. Sankaranarayanan, S. Velusamy, M. Stan, C. L, and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *Journal of Instruction Level Parallelism*, vol. 7, 2005.

[7] Y. Han and I. Koren, "Simulated Annealing Based Temperature Aware Floorplanning," *Journal of Low Power Electronics*, vol. 3, 2007.

[8] A. Gupta, N. Dutt, F. Kurdahi, K. Khouri, and M. Abadir, "LEAF: A System Level Leakage-Aware Floorplanner for SoCs," *Proc. of the 2007 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 274–279, Jan. 2007.

[9] C.-H. Tsai and S.-M. Kang, "Cell-level placement for improving substrate thermal distribution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 253–266, Feb. 2000.

[10] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, May 2006.

[11] A. Ajami, K. Banerjee, and M. Pedram, "Modeling and analysis of nonuniform substrate temperature effects on global ULSI interconnects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 849–861, June 2005.

[12] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, Jan. 1948.

[13] J. Black, "Electromigration - a brief survey and some recent results," *Electron Devices, IEEE Transactions on*, vol. 16, no. 4, pp. 338 – 347, apr 1969.

[14] T.-C. Chen and Y.-W. Chang, "Modern floorplanning based on b*-tree and fast simulated annealing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 4, pp. 637 – 650, april 2006.

Bibliography

[1] R. Mahajan, C. P. Chiu, and G. Chrysler, "Cooling a microprocessor chip," *Proceedings of the IEEE*, vol. 94, pp. 1476–1486, 2006.

[2] "International technology roadmap for semiconductors (itrs)," http://www.itrs.net/, August 2011.

[3] C. liu, J. Su, and Y. Shi, "Temperature-aware clock treee synthesis considering spatiotemporal hot spot correlations," *IEEE*, pp. 107–113, 2008.

[4] M. Cho, S. Ahmed, and D. Z. Pan, "Taco: Temperature aware clock-tree optimization," *IEEE*, pp. 581–586, 2005.

[5] A. Chakraborty, K. Duraisami, A. Sathanur, P. .Sithambaram, L. Benini, A. Macii, E. Macii, and M. Poncino, "Dynamic thermal clock skew compensation using tunable delay buffers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, No. 6, pp. 639–649, 2008.

[6] A. Gupta, N. D. Dutt, F. J. Kurdahi, K. S. Khouri, and M. S. Abadir, "Thermal aware global routing of vlsi chips for enhanced reliability," *9th International Symposium on Quality Electronic Design*, vol. 8, p. 6, 2008.

[7] K. Lu and D. Z. Pan, "Reliability-aware global routing under thermal considerations," *1st International Symposium on Quality Electronic Design-Asia*, vol. 1, p. 6, 2009.

[8] "Hotspot research team," http://lava.cs.virginia.edu/HotSpot/people.htm, August 2011.

[9] C. S. department at University of Virginia, "Hotspot homepage," http://lava.cs.virginia.edu/HotSpot/index.htm, August 2011.

[10] W. Liu, "Power and thermal management of system-on-chip," Ph.D. dissertation, Danmarks Tekniske Universitet, 2011.

[11] N. H. E. Weste and D. M. Harris, *CMOS VLSI design*, M. Hirsch, Ed. Addison-Wesley, 2011.

[12] Y. Zhan, S. V. Kumar, and S. S. Sapatnekar, "Thermally aware design," *Foundations and Trends in Electronic Design Automation*, vol. 3, pp. 255–370, 2008.

[13] C. J. M. Lasance, "Thermally driven reliability issues in microelectronic systems: Status quo and challenges," *Microelectronics Reliability*, vol. 43, no. 12, pp. 1969–1974, 2003.

[14] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebenheur, "Thermal performance challenges from silicon to systems," *Intel Technology Journal*, vol. Q3, 2000.

[15] F. d'Heurle, "Electromigration and failure in electronics: An introduction," *Proceedings of the IEEE*, vol. 59, no. 10, pp. 1409–1418, 1971.

[16] J. Black, "Electromigration - a brief survey and some recent results," *IEEE Transactions on Electron Devices*, vol. 4, pp. 338–347, 1969.

[17] D. K. Schroder and J. A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufactoring," *Journal of Applied Physics*, vol. 94, no. 1, pp. pp 1–18, 2003.

[18] D. Atienza, G. D. Micheli, L. Benini, J. Ayala, P. D. Valle, M. Debole, and V. Narayanan, "Reliability-aware design for nanometer-scale devices," *Proc. of the 2008 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 549–554, 2008.

[19] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *Journal of Instruction-level Parallelism*, 2005.

[20] Y. Han and I. Koren, "Simulated annealing based temperature aware floorplanning," *Journal of Low Power Electronics*, vol. 3, 2007.

[21] A. Gupta, N. Dutt, F. J. Kurdahi, K. S. Khouri, and M. S. Abadir, "Leaf: A system level leakage-aware floorplanner for socs," *Proc. of the 2007 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 274–279, Jan. 2007.

[22] "Example floorplan," http://sequoia.ict.pwr.wroc.pl/~witold/aiarr/2007_projekty/vlsi/, September 2011.

[23] J. E. HarlowIII", "Overview of popular benchmark sets," *IEEE Design and Test of Computers*, pp. 15–17, 2000.

[24] W. Liu, "Temperature dependant wire delay modelling at the floorplanning stage," Institute of Mathematical Modelling at Danmarks Tekniske Universitet, Tech. Rep., 2011.

[25] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Predictable routing," *Proceedings of the 2000 IEEE/ACM International Conference on Computer Aided Design*, pp. 110–113, 2000.

[26] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, 6, pp. 1087–1092, 1953.

[27] R. Carr, "Simulated annealing," http://mathworld.wolfram.com/SimulatedAnnealing.html, July 2011.

[28] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical Science*, vol. 8, No. 1, pp. 10–15, 1993.

[29] "Simulated annealing survey," http://www.cs.sandia.gov/opt/survey/sa.html, July 2011.

[30] L.-T. Wang, Y.-W. Chang, and K.-T. Cheng, *Electronic Design Automation: Synthesis, Verification, and Test (Systems on Silicon)*. Morgan Kaufmann, 2009.

[31] A. Ajami, K. Banerjee, and M. Pedram, "Modeling and analysis of nonuniform substrate temperature effects on global ulsi interconnects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 849–861, 2005.

[32] W. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, 1948.