Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA ELETTRONICA, DELLE TELECOMUNICAZIONI E TECNOLOGIE DELL'INFORMAZIONE

Ciclo XXVI

Settore Concorsuale di afferenza: 09/E3 Elettronica

Settore Scientifico disciplinare: ING-INF/01

TEMPERATURE VARIATION AWARE ENERGY OPTIMIZATION IN HETEROGENEOUS MPSOCS

Presentata da: Mohammadsadegh Sadri

Coordinatore Dottorato	**Relatore**
Prof. Alessandro Vanelli Coralli	Prof. Luca Benini

Esame finale anno 2014

Temperature Variation Aware Energy Optimization in Heterogeneous MPSoCs

Mohammadsadegh Sadri

DEI - Dpt. of Electrical, Electronic and Information Engineering

University of Bologna

A thesis submitted for the degree of

Doctor of Philosophy

Supervisor: *Prof. Luca Benini*, Submission Date: *March - 2014*

I would like to dedicate this thesis to my loving family.

Abstract

Thermal effects are rapidly gaining importance in nanometer heterogeneous integrated systems. Increased power density, coupled with spatio-temporal variability of chip workload, cause lateral and vertical temperature non-uniformities (variations) in the chip structure. The assumption of an uniform temperature for a large circuit leads to inaccurate determination of key design parameters. For this reason, significant design margins are taken to ensure safe operation of the device. To improve design quality, we need precise estimation of temperature at detailed spatial resolution which is very computationally intensive. Consequently, thermal analysis of the designs needs to be done at multiple levels of granularity.

To further investigate the flow of chip/package thermal analysis we exploit the Intel Single Chip Cloud Computer (SCC) and propose a methodology for calibration of SCC on-die temperature sensors. We also develop an infrastructure for online monitoring of SCC temperature sensor readings and SCC power consumption. We create a power and a thermal model for Intel SCC. The accuracy of the models is verified through a set of practical experiments by performing direct comparisons with the real hardware measurements. Considering the importance of Dynamic Voltage and Frequency Scaling (DVFS) technique for reducing power consumption in multi-core systems, we also quantify the effect of frequency scaling on the performance and energy efficiency of parallel workloads in many-core platforms.

Having the thermal simulation tool in hand, we propose MiMAPT, an approach for analyzing delay, power and temperature in digital integrated circuits. MiMAPT integrates seamlessly into industrial Front-end and Back-end chip design flows. It accounts for temperature non-uniformities and self-heating while performing analysis. MiMAPT performs thermal analysis at RT and gate-level with multiple scales of resolution and speed. It considers non-uniform shapes of on-die units. We demonstrate the capability of MiMAPT in temperature variation aware delay/power analysis using a widely-used digital IP block implemented in 65nm and 40nm technology nodes.

Furthermore, we extend the temperature variation aware analysis of designs to 3D MPSoCs with Wide-I/O DRAM. We improve the DRAM refresh power by considering the lateral and vertical temperature variations in the 3D structure and adapting the per-DRAM-bank refresh period accordingly. We develop an advanced virtual platform which models the performance, power, and thermal behavior of a 3D-integrated MPSoC with Wide-I/O DRAMs in detail. On this platform we run the Android OS with real-world benchmarks to quantify the advantages of our ideas.

Moving towards real-world multi-core heterogeneous SoC designs, a reconfigurable heterogeneous platform (ZYNQ) is exploited to further study the performance and energy efficiency of various CPU-accelerator data sharing methods in heterogeneous hardware architectures. A complete hardware accelerator featuring clusters of OpenRISC CPUs, with dynamic address remapping capability is built and verified on a real hardware. This platform provides us with the possibility of performing further research on mechanisms for CPU-accelerator memory sharing as well as improving energy efficiency by considering on-die spatio-temporal temperature variations and scheduling processing tasks to accelerators accordingly.

Contents

Contents **iv**

List of Figures **vii**

List of Tables **x**

1 Introduction **1**

1.1 Thesis Outline 8

2 Thermal Models, Their Creation and Validation **10**

2.1 Thermal Simulation Principles 11

2.2 A Survey on Available Academic Thermal Models 12

2.3 Intel Single Chip Cloud Computer 13

2.4 Thermal Simulation for Intel SCC 14

2.4.1 A Power Model for Intel SCC 15

2.4.1.1 SCC Power Measurement 17

2.4.1.2 Power Model for SCC CPU Cores 18

2.4.2 SCC Thermal Sensors 20

2.4.3 Thermal Sensors Calibration 23

2.4.4 Hotspot Thermal Model for Intel SCC 23

2.4.5 Validation of Thermal Model 24

2.5 Quantifying the impact of frequency scaling on energy efficiency of SCC 28

2.5.1 Tailored Benchmarks 28

2.5.2 Experimental Evaluation 30

3 Temperature Variation Aware Delay, Power and Thermal Analysis 34

3.1 The Importance of Temperature Variation Aware Analysis 34

3.2 MiMAPT Architecture 37

3.2.1 Temperature Variation Aware Delay and Power Estimation . . 37

3.2.2 Multi-scale Thermal Simulation at RT and Gate Level 39

3.2.2.1 RT Level Hotspot Detection 41

3.2.2.2 Gate Level Hotspot Detection 43

3.2.3 Merged Virtual Chip Analysis 46

3.3 Thermal Impact Exploration 46

3.4 Adaptive Hotspot Detection at RTL 58

3.5 Example MiMAPT Operation 59

4 Temperature Variation Aware Energy Optimization in 3D MPSoCs 66

4.1 Related Research Works 67

4.2 The ESL Virtual Infrastructure 69

4.2.1 DRAM Model 71

4.2.2 CPU Model 73

4.2.3 Thermal Model 74

4.2.4 Refresh Governor 77

4.2.5 Thermal Governor 77

4.2.6 Sampling Governor 79

4.3 Experimental Results 80

4.3.1 Temperature Variation Aware Bank-wise Refresh 80

4.3.2 Thermal Controller 80

4.3.3 Adaptive Sampling 83

4.4 Hardware Acceleration of Thermal Simulation 84

5 Energy and Performance Evaluation of CPU-Accelerator Memory Sharing Using Xilinx ZYNQ 86

5.1 Related Research Works 87

5.2 Xilinx ZYNQ Heterogeneous System-on-Chip 89

5.3 Infrastructure for Energy and Performance Quantification of CPU-Accelerator Data Sharing 91

5.3.1 Hardware 92
5.3.2 Software 94
5.3.3 Power Measurement 95
5.3.4 CPU-Accelerator Data Sharing Methods 96
5.3.5 Experimental Results 97

6 Temperature Variation Aware Hardware Acceleration in Heterogeneous MPSoCs **103**
6.1 OpenRISC Based Heterogeneous Multi-Core SoC 103
6.1.1 Hardware Architecture 104
6.1.2 Address Map and Dynamic Address Remapping 112
6.1.3 Practical Implementation 116
6.2 Future Work 116
6.2.1 Exploration of CPU-Accelerator Address Sharing Methods . . 118
6.2.2 Implementation of Thermal Sensors 119
6.2.3 Temperature Aware Assignment of Tasks to Accelerators . . . 120

7 Conclusions **122**

Publications **125**

References **127**

List of Figures

2.1 Block diagram of of Intel SCC. 13

2.2 Die photo of of Intel SCC. 14

2.3 Core power consumption with number of active cores in each cluster for each of the workloads. Dots are measured data; Lines are the result of linear fitting. 17

2.4 Core power consumption for each workload for different frequency values. Dots are measured data values. Lines are the result of linear fitting. 18

2.5 Changes of CPI for different values of frequency and different workloads. 19

2.6 Power consumption of DRAM controllers as a function of frequency and number of active cores running. 19

2.7 Power consumption of one CPU core based on CPI values and running frequency. Dots are measured values. Lines are the model output. . . 21

2.8 Raw thermal sensor values for two different tests. 22

2.9 SCC thermal sensor readings with the change in total chip power consumption. 22

2.10 Comparison of measured temperature values with Hotspot ones when half of the chip is at idle state and the other half is at full load. 25

2.11 Statistical distribution of error values. For each error value on x axis, the value on y axis shows the percentage of points with error value higher than the specified error value. 26

2.12 Temperature readings by the thermal sensor located on the main-board of SCC and one of on-die thermal sensors. 26

2.13 Sensitivity of the Intel SCC benchmarks to frequency scaling. 32

3.1 MiMAPT Block Diagram 40
3.2 Generation of power map for thermal simulation at RTL 43
3.3 Increasing spatial resolution of thermal floorplan for the area of interest. Example, one FFT unit implemented in 40nmLVT running at 800MHz. 45
3.4 The operation of MiMAPT during merged virtual chip analysis flow. . 47
3.5 Available characterized corners for 40nm and 65nm nodes. 49
3.6 Dynamic and static power density values and delay of 4 implemented Chips, for the main operating corners. All numbers are normalized to the values of the 65nm HVT chip at `WCL` corner for which static power density is 2×10^{-4} W/cm^2, dynamic power density 2.79×10^{-3} W/cm^2/MHz and maximum clock frequency 61 MHz. 50
3.7 Physical location of first 10 timing critical paths of the design for different uniform temperature values. First path is shown in black. 55
3.8 Location of first 10 timing critical paths for different temperature maps and supply voltages. First critical path is in black. 58
3.9 hotspot detection methods comparison (Adaptive vs. TH only) 59
3.10 Total estimated power for each of the *TF*s at RT and Gate level. The chip contains 16 instances of FFT unit. Red circles show *TF*s for which a suspicious block is detected. 60
3.11 Sample temperature gradients obtained during MiMAPT analysis on the sample chip. 61
3.12 Physical coordinates of hottest point of the chip when TVA is disabled (shown with n letter), TVA is active and temperature map is used (y), TVA is active and averaged die temperature is used (v). 64

4.1 Virtual Platform with Thermal Control Loop 70
4.2 Flow of one Co-Simulation step 71
4.3 3D stack-up of the virtual MPSoC 75
4.4 Geometries used for thermal model: (a) DRAM die (b) Core die. . . . 76
4.5 Refresh periods of Channel 3, (a) Bank-wise off (b) Bank-wise on (SmartBench). 81
4.6 Change in sampling interval relative to temperature 84

5.1 A block diagram representing important elements of the Xilinx ZYNQ device. 89
5.2 Block diagram of the developed hardware. 93
5.3 Processing bandwidth comparison of acceleration methods. Image size sweeps from 4 KB to 2048 KB. 98
5.4 Averaged power consumption of major system blocks for each processing method. 100
5.5 Energy consumed for processing of one byte of data for each processing method. 101
5.6 Processing bandwidth comparison of ACP and HP0 accelerators at the presence of background traffic. 102

6.1 Developed architecture containing the ARM host (ZYNQ PS) connected to the L2 memory and two cluster modules. Each cluster module contains 4 OpenRISC cores. The RAB Table is responsible for translating addressese before passing them to the HP0 port of ZYNQ PS. 105
6.2 The implemented architecture for one OpenRISC cluster module. It contains a cluster core unit and all of the required AXI interconnects as well as local cluster peripherals. The AXI multi-port central DMA is and its connection to the second port of TCDM memories is also depicted. 106
6.3 Architecture of one Cluster Core unit, which contains 4 instances of OpenRISC cores, 4 instances of demultiplexer unit, the logarithmic interconnect and tightly coupled memory blocks. 110
6.4 The address map of the system as seen by the components inside the accelerator. 115
6.5 The layout of the final routed design on Xilinx XC7Z054 device. The building elements of each of the major blocks in the design are highlighted with different colors. 117
6.6 Tap delay circuit built using LUTs for temperature sensing. 120
6.7 Sample waveforms of tap delay circuit operation. 120

List of Tables

1.1 Delay and Power Densities (P.D.) for a 8-bits counter using 40nmLP RVT cells. (Delay in ns and Power Density in W/cm^2) 3

2.1 Power break-down of different units in Intel SCC for two different extreme cases. 15
2.2 Workloads used for stressing SCC. (One cache line of SCC is 32 bytes). 16
2.3 Important Hotspot thermal model configuration parameter values for Intel SCC. 24
2.4 Comparison of real measurements with model computed data for 10 diifferent tests. Ambient temperature $T_{amb} = 40$ °C. 27
2.5 Benchmark Categorization 29

3.1 TSMC 40nmLP and 65nmLP technology nodes 48
3.2 Area, sequential to combinational cells ratio and number of clock buffers, for each of 4 implemented chips. (Area: mm^2) 49
3.3 Sensitivity of static/dynamic power and delay to changes in temperature for fixed supply voltage. (Change over 1 °C, relative to final value at 125 °C in percents. Static power absolute change value is in (W/Cm^2)/C) 51
3.4 Estimated delay/power values for BC corner when using characterized library compared to library interpolation 52
3.5 Changes in power and delay with temperature sweep from $0C$ to $100C$. A unique temperature is assumed for entire die. 54

3.6 Relative error values for estimated power, minimum clock period and physical location of first timing critical path, when considering non-uniform on-die temperature compared to assumption of uniform averaged temperature for entire die. 57
3.7 Definition of symbols representing execution times of different tasks done during analysis of one *TF*, and their average value for the example chip. 63
3.8 Execution times comparison: MiMAPT vs fine-grain. (Time in seconds$\times 10^3$) 64

4.1 Refresh rates of DRAM banks vs. temperature 73
4.2 t_{sim} vs. maximum chip temperature 79
4.3 bank-wise refresh vs normal method. Thermal controller off, CPU cores @1.25 GHz, All values are averaged. 82
4.4 Impact of each control knob on the temperature and performance of the chip (Benchmark: smartbench) 82
4.5 Key system statistics for each thermal controller test. ($K_P = 350$, $K_I = 1.17$, $K_D = 0.2$) 83

6.1 Resource utilization of XC7Z045 APSoC for a cluster containing 8 OpenRISC CPU cores and all required glue logic. 118

List of Algorithms

1 Apply temperature map to design for power/timing analysis 38
2 Generation of power map for thermal simulation at RTL 42
3 Adaptive Hotspot Detection 44
4 Multi-granularity Thermal Simulation 45

Chapter 1

Introduction

High power densities of today's integrated circuits lead to on-chip thermal hotspots which can compromise chip functionality [62]. The volumetric power density of a 20nm FinFET device is on the order of 10 TW/cm^3 [117]. This power wall slows down the progress toward higher transistor densities and faster clocks while keeping reliability at acceptable levels [46]. Consequently, researchers and CAD vendors are developing solutions and tools at different levels of abstraction (such as software [35], micro-architecture [22], floorplanning and cell placement [49]) to facilitate prediction and identification of thermal hazards.

To further complicate matters, operating temperature has significant impact on key design characteristics such as speed and leakage power [89]. Consequently, to ensure correct functionality for all of the targeted environmental conditions, work-loads and self-heating effects, Front-end and Back-end flows are needed to account for many operating points, tight design constrains with large safety margins [37]. Moreover the design of essential components of the system (e.g. heat-sink and PCB) will be highly affected by these constraints [114]. These directly translate into additional costs [85]. As a result, a big percentage of available resources are consumed to only guarantee the safety of chip operation in situations that may never happen in practice. This is where design-time thermal analysis can be useful. Conceptually, taking temperature variation into account while doing delay/power analysis, allows lowering safety margins without compromising design reliability. This is done by filtering out the constraints that never happen in practice and satisfying only those which are really needed.

Spatial and temporal variability of chip workload results in non-uniform on-chip power density and thus localized temperature variations. Considering Intel Single-chip Cloud Computer (SCC) many-core platform [60] as an example, the power consumption of on-die CPU cores, while running Linux OS, changes more than $17X$ from idle to full-load state. Accounting for the fact that each group of cores can have their own different supply voltage and running clock, the same level of variation can be seen spatially, in per-core power consumption. For today's high-end multi-cores, temperature gradients of more than 15 °C can be seen across different on-die units [99]. As a result, considering the serious impact of temperature on device characteristics such as leakage power and delay (which change non-linearly with temperature), and abrupt temperature variations across die area, the assumption of an averaged uniform temperature value for entire die does no more lead to practically valid delay/power estimations. In fact, today's mainstream design flows assume a uniform temperature value for each operating corner during design synthesis and evaluation and a quantification of the error caused by this assumption is currently missing.

To clearly illustrate the idea, we assume that an 8-bits counter is the only timing critical path of a sample design implemented using TSMC 40nm low power (LP) standard cell library [126]. As shown in table 1.1, for $VDD = 0.81$ V, considering the full temperature range (0 °C to 125 °C), the maximum running frequency of the design will be 457 MHz (period 2.18 ns). However, if through thermal simulations we show that the temperature of timing critical path area never goes below 50 °C, the chip can be safely clocked at more than 640 MHz (period 1.54 ns). The result of removing this unnecessary safety margin is $1.4X$ improvement in performance. For $VDD = 1.21$ V however, the impact of temperature on delay is not significant.

For $VDD = 1.21$ V, if we suppose that half of the chip area is covered by circuits similar to the counter, and total chip area is 560 mm^2 (similar to [60]), then considering full temperature range, the total power consumption changes by more than 21 Watts (from -40 °C to 125 °C). However, if through thermal simulations we show that the temperature of these areas never grows over 50 °C, then the change in power consumption will be lower than 11 W. This reduction in safety margins allows us to use lighter cooling solutions as well as lower priced power supply components. For $VDD = 0.81$ V the impact of temperature on power is not significant.

The increase in power density with temperature translates directly into increase in localized temperature which again results in change over delay and power. The created

Table 1.1: Delay and Power Densities (P.D.) for a 8-bits counter using 40nmLP RVT cells. (Delay in ns and Power Density in W/cm^2)

VDD	Parameter	Temperature			
		-40	0	50	125
0.81	Clk Period	2.1872	1.7269	1.5486	1.1450
	P.D.@slowest Clk	11	12	12	12
1.21	Clk Period	0.1993	0.2029	0.2080	0.2138
	P.D.@slowest Clk	270	279	289	307

iterative loop moves the actual operating condition of transistors to completely different points than initially estimated ones. This highlights the importance of considering self-heating effects while performing design validation.

Moreover, considering high power densities, precise localization of hotspots at gate level is necessary for many high-performance designs. This however requires an extremely detailed spatial resolution for power/thermal simulation which is very computationally intensive and almost impossible for large designs. This is where multi-scale analysis techniques can help. Basically, power and temperature estimation can be done at low resolution where it provides satisfactory level of accuracy at a high level of speed. Resolution (and computational effort) should be increased only for areas of interest, where hotspots are likely. The challenge lies in avoiding false negatives (i.e. missing hot spots) while minimizing false positives to achieve significant speedups w.r.t. fine-grained (gate level) thermal analysis.

The first step to tackle the aforementioned problems is to be able to practically estimate the temperature distribution across a given hardware platform. Thus, we first focus on the procedure of performing thermal simulation for a given chip/package design. We study the principles of developing thermal models for integrated circuits. Then we investigate the operation of various thermal models such as Hotspot [61] and 3D-ICE [120]. To further describe the flow of chip/package thermal simulation, we create a thermal model for the Intel SCC [4]. Then we use the actual real hardware and tune our thermal model. Finally we validate the accuracy of the proposed thermal model through practical experiments. Towards this, we study the architecture of temperature sensing hardware instantiated inside Intel SCC as well as development of an infrastructure for online extraction of sensor readings from the hardware [2]. We illustrate a method for calibration of thermal sensors [1]. We further study the effect of frequency scaling on the performance and energy efficiency of parallel workloads running on many-core platforms [2].

Having the thermal analysis tool in hand, we then propose MiMAPT (Micrel Multi-scale Analyzer for Power and Temperature), a tool capable of performing temperature variation aware delay/power and thermal analysis at RT and gate level with multiple scales of resolution and speed. The tool starts coarse-grained transient power/thermal analysis at RTL for the set of user-defined workloads. It considers non-uniform shapes of on-die units during analysis. MiMAPT switches to accurate gate level simulation only when a likely hot-spot is suspected. At gate level it performs iterative power/thermal simulation while refining spatial resolution of the thermal floorplan just for the areas which are suspected to contain hotspots. In addition, MiMAPT accounts for on-die temperature variations and non-uniformities as well as self-heating effects while performing delay/power and temperature analysis at gate level.

Furthermore, MiMAPT provides the designers with the possibility of early design evaluation when the design is available as stand-alone IP blocks but not as the final implemented chip. In fact, MiMAPT is capable of reading IP blocks and their workloads separately, and merging all IP units into a virtual chip with each unit placed at its user-specified coordinates. On the virtual chip, MiMAPT performs thermal analysis at RT and gate level, as well as temperature variation aware power/delay estimation. MiMAPT handles all of the necessary translations in the provided RT and gate level data files for each IP block automatically. In the special cases that an IP block is not yet available, or is completely a black-box, MiMAPT allows the developer to define a dummy block with an averaged power density to represent the missing unit during chip-wide analysis. When the IP is available only at gate level but not at RTL, MiMAPT can use the already available information of the IP along with the rest of the design during its multi-scale analysis.

The tool integrates seamlessly into major industrial Front-end and Back-end chip design flows. It exploits standard logic simulation engines to obtain circuit switching activities required for design power estimation at RT and Gate level. It parses standard cell library formats, as well as widely used file formats describing chip layout and placement information. It is also capable of processing and using reported delay/power traces by major Front-end and Back-end tools. Furthermore, MiMAPT is designed so that it can be easily connected to an external thermal simulation engine.

Thanks to MiMAPT we explore and model the sensitivity of key design parameters to temperature in two recent technology nodes (65nm [127] and 40nm [126]), using real-life industrial libraries. We further demonstrate the importance of considering on-die temperature non-uniformities to obtain accurate temperature maps for the chip. The principal ideas behind MiMAPT can be also applied to 3D integrated circuits.

Heterogeneous 3D integrated systems with stacked DRAMs are a promising solution to squeeze more functionality and storage bits into an ever decreasing volume. Unfortunately, with 3D stacking, the challenges of high power densities and thermal dissipation are exacerbated. Energy and thermal dissipation are limiting the efficiency of today's applications performance on smartphones as well as high-end servers. The thermal issues of 3D ICs cannot be solved by tweaking the technology and circuits alone. In fact, a 3D stacked SoC aggravates the thermal crisis and forces enhancements in the architecture and memory organization as well as detailed DVFS control for CPUs and DRAMs. More than 40% of the system energy in existing platforms is consumed by DRAMs [90].

Early detection of architectural shortcomings and thermal hazards is crucial to the design of sub-20-nm 3D chips. For instance, current microprocessor architectures are inefficient for running datacenter workloads mainly because of the mismatch between the workload characteristics and the organization of the memory subsystem [45]. Consequently, the detailed analysis of the memory subsystem is very important as it unveils possible bottlenecks and issues which impact the system energy and efficiency.

Therefore, in this thesis, we perform an in-depth study on performance, timing, power and leakage of 3D stacked Wide-I/O DRAMs and track their key parameters with spatial and temporal variations in temperature. Through careful evaluation of temperature distributions in a 3D IC with Wide-I/O DRAMs, we propose architectural enhancements for the DRAM subsystem as well as thermal management solutions which improve energy consumption. We consider the lateral and vertical variation in temperature of the 3D DRAM dies and refresh each of the DRAM banks at a separate rate according to its own temperature.

In order to assess and quantify the advantages of our proposed ideas, we build a suitable virtual infrastructure which considers all key characteristics of a 3D MPSoC with Wide-I/O DRAMs in detail. Our virtual platform uses *Transaction Level Models* (TLM), since they are well suited for fast system-level simulation and exploration of designs. Moreover, *Approximately Timed* (AT) TLM2.0 models provide a sophisticated balance between accuracy and execution speed of simulations.

The analysis of the memory subsystem requires a timing accurate behavior of the CPU cores. The *gem5* [21] architecture level full-system simulator is selected for this purpose, since it models system operations at various levels of detail, balancing simulation speed and accuracy. It generates realistic traces of memory accesses, which can be replayed very fast inside the TLM environment. Detailed O3 CPU model of *gem5* resembles the operation of a real hardware. As an outcome, the statistics generated by the mentioned models can be utilized for estimation of power and temperature with acceptable level of accuracy.

Our infrastructure integrates the *gem5*, TLM2.0 and our developed power and thermal models, as well as various thermal, performance and operation management modules into a unified working set. Further speedup of simulation is obtained by adaptively adjusting the sampling interval according to the chip temperature and accelerating the thermal simulation. Our framework allows us to explore different thermal control and DRAM refresh management strategies and assess the behaviour of the chip stack thermal profile and its energy dissipation under the real-world workloads.

As the energy efficiency requirements (e.g. GOPS/W) of silicon chips are growing exponentially, computer architects are seeking solutions to continue application performance scaling. One emerging solution is to use specialized functional units (accelerators) at different levels of a heterogeneous architecture. These specialized units cannot be used as general-purpose compute engines. However, they provide enhanced execution speed and power efficiency for their specific computational workloads [26]. There exist numerous applications for accelarators in both of the embedded and high performance computing markets. Examples include video processing [104], software-defined radio [34], network traffic management [79], DNA computing [77] and fully programmable hardware acceleration platforms [96].

In the final part of the thesis, we focus on heterogeneous MPSoCs which contain a group of hardware accelerator units capable of executing computational tasks in parallel with the host CPU. We study the idea of accounting for on-die temperature non-uniformities while assigning computational tasks to available hardware accelerators. We build a complete heterogeneous reconfigurable platform containing a multi-core host CPU as well as a cluster of hardware accelerators. We first study the possible mechanisms of assigning computational tasks to hardware accelerators by the host CPU. Especially, we focus on CPU-Accelerator memory sharing mechanisms and study the power and performance efficiency of each method through practical tests. Then we illustrate the idea of implementing thermal sensors on various spatial coordinates of the fabric which contains the accelerators cluster. Finally we focus on the problem of scheduling the processing tasks to accelerator units based on the obtained temperature readings.

Efficient sharing of data in a heterogeneous MpSoC which contains different types of integrated computational elements is a challenging task. Especially when private caches of CPU cores and dedicated memory of accelerators are used to store local copies of data in a hierarchical memory structure, it is crucial to ensure that every processing element has a consistent view of the shared memory space [122], [78]. The Accelerated coherency port (*ACP*) [121] was developed by ARM® as a hardware solution to enable hardware accelerators to issue coherent requests to the CPU subsystem memory space [67].

The memory used for sharing data between the host CPU and hardware accelerator is primarily indicated by *virtual address* values allocated by the host CPU. These addresses need to be converted into equivalent physical addresses before performing any access to the actual physical memory. At CPU side, this address conversion is automatically done by the MMU however, for hardware accelerator the CPU needs to perform address conversion explicitly and send the physical address to the accelerator. This conversion is simple if the shared object is a small chunk of continuous memory however, as soon as the size and complexity of the shared data object increases and it spans several non-continuous physical regions, the simple method can no more be used. To tackle this problem, we propose a new CPU-accelerator address passing method in which the accelerator receives the virtual address from the CPU directly, and then uses a specialized look-up table to obtain the equivalent physical address. This scenario resembles the operation of a *System-MMU* such as the one disscussed in [66].

We need to assess our ideas on a real-world multi-core heterogeneous platform. However, creating a TLM platform resembling the operation of such complex piece of hardware with acceptable levels of simulation speed and accuracy is impossible.

As a result we move to a real-world platform to continue our research. Xilinx ZYNQ all-programmable SoC [140] provides the designers with an ARM *Cortex-A9 MPCore* subsystem along with a high performance DRAM memory controller and various peripherals. It also implements a complete FPGA fabric. The CPU subsystem and FPGA are connected through AXI [68] interfaces which allow the logic on the fabric to perform cache coherent accesses to the memory space of the cpu subsystem through the *ACP* or directly perform accesses to the DRAM.

We use the ZYNQ device and build a complete infrastructure to evaluate the performance and energy efficiency of different processor-accelerator memory sharing schemes as well as our ideas on temperature variation aware hardware acceleration.

1.1 Thesis Outline

We first describe the methodology of performing thermal simulation for ICs in Chapter 2. Primarily, we focus on principles of thermal simulation (Section 2.1). Section 2.2 contains a brief survey on available widely used thermal models. Then we introduce our research vehicle: Intel SCC (Section 2.3). We focus on developing a power model (Section 2.4.1) and a thermal model (Section 2.4.4) for Intel SCC. We describe the operation of SCC integrated on-die thermal sensors (Section 2.4.2) and the required steps for their calibration (Section 2.4.3). Next, we perform a practical comparison between the temperature values reported by SCC thermal sensors and the obtained temperatures from our thermal model (Section 2.4.5). Finally, we provide quantifications on the effect of frequency scaling in many-core platforms on the performance and energy efficiency of parallel applications (Section 2.5).

Chapter 3 is dedicated to detailed description of our RT and Gate Level, Adaptive and Multi-scale delay, power and thermal analysis engine: MiMAPT. First in Section 3.1 we illustrate the importance of considering temperature variation while doing design analysis. The overall architecture of MiMAPT is described in Section 3.2. A set of practical examples to show the effect of on-die temperature non-uniformities on key design parameters is demonstrated in Section 3.3. Then we perform an evaluation on the accuracy of our adaptive hotspot detection algorithm at RT level (Section 3.4). Finally, through an example we show the full functionality of MiMAPT (Section 3.5). This contains the multi-scale analysis capability of MiMAPT working jointly with its temperature variation aware power/delay estimation engine.

Chapter 4 describes the TLM modeling of 3D MPSoCs with stacked DRAMs. It also illustrates our ideas regarding considering the lateral and vertical temperature non-uniformities in the 3D chip to optimize the energy consumption of DRAM memories. To perform our assessments, we develop a detailed virtual platform containing TLM models for 3D Wide-I/O DRAM memories. Section 4.2 explains this in detail. To improve the energy consumption of the 3D MPSoCs a set of governor units are implemented. Each governor is responsible for monitoring and managing one of the key parameters of the system. Sections 4.2.4 to 4.2.6 describe these modules in detail. Finally we quantify the effectiveness of the developed ideas through a set of experiments (Section 4.3).

Considering the fact that for real-world's modern complex multi-core heterogeneous platforms, creating TLM models for the entire hardware with acceptable level of speed and accuracy is impossible, we then focus on using re-configurable heterogeneous platforms like Xilinx ZYNQ to implment real heterogeneous systems and to bring the possibility of continuing our research on a real test platform.

In Chapter 5, we first discuss the architecture of Xilinx ZYNQ (Section 5.2). We then focus on measuring the performance and energy consumption of each of the available methods for sharing chunks of data between CPU and accelerator (Section 5.3).

Chapter 6 demonstrates the infrastructure that we develop on the Xilinx ZYNQ device to further evaluate our ideas regarding CPU-Accelerator data and address sharing, as well temperature variation aware assignment of computation tasks to hardware accelerators. In this chapter, we desribe our fully functional OpenRISC based accelerator. The details on the implemented cluster are presented in Section 6.1.

We propose two different ideas as our future research directions; The idea of implementing a collaborative system MMU which enables the host CPU to pass the virtual address of a data structure to the accelerator directly is disscussed in Section 6.2.1. In Section 6.2.2 we show possible ways of implementing thermal sensors on the FPGA fabric directly. Finally, in Section 6.2.3 we describe the idea of assigning computational tasks to hardware accelerator blocks according to their current temperature state.

Chapter 2

Thermal Models, Their Creation and Validation

Upcoming many-cores platforms stress the limits of Moores law. Indeed high performance rush translates in high power densities, that combined with high spatial parallelism and workload variations produces non-uniform power dissipation that translate in non-uniform silicon die thermal map. This leads to degradation, acceleration of chip aging and increase in cooling costs. To help designers in studying and counteracting this raising issue thermal modeling tools has been widely studied in the recent years, nonetheless leading industries have started to deliver the firsts many-core prototypes to push researchers toward creating solutions for the incoming challenges.

In this Chapter, we first study the principles of chip/package thermal simulation briefly (Section 2.1). Then, we introduce the well-known available thermal modeling tools (Section 2.2). Afterwards, in Section 2.3, we introduce our research vehicle: the Intel Single Chip Cloud Computer (SCC). We develop a thermal model for Intel SCC (Section 2.4) and we verify the validity of our model in practice (Section 2.4.5).

Dynamic voltage and frequency scaling (DVFS) is one of the most commonly utilized techniques for reducing power consumption of chips. Recent research has developed efficient DVFS techniques based on characterizing on-chip/off-chip workloads [33], identifying application phases with a high number of stall cycles [76], or using machine learning techniques to adapt to changing workload [40]. The common goal in these approaches is reducing the performance overhead of operating at lower frequencies, as DVFS may incur severe slow-downs. These techniques improve the energy efficiency for the current single-core or multi-core systems with a few number of cores; however, they do not capture the unique performance-power trade-offs in many-core systems. As a result, in the final part of this chapter, we study the effect of frequency scaling on the performance and energy efficiency of parallel workloads running on many-core platforms (Section 2.5).

2.1 Thermal Simulation Principles

The classical Fourier heat diffusion model is used to model heat transfer through chip and cooling package. IC chip and cooling packages are spatially discretized into discrete three-dimensional thermal elements. Compact heat transfer equations are then derived and solved using numerical methods to characterize the thermal profile of IC chip and cooling package. Both steady-state and dynamic analysis methods have been developed for full-chip IC thermal analysis. Compared with steady-state thermal analysis, dynamic thermal analysis is much more challenging. The steady-state classical Fourier model is characterized by the following equation:

$$\nabla \cdot (K \nabla T) + q_{vol} = 0 \tag{2.1}$$

where K is the thermal conductivity, T is temperature, and q_{vol} is the volumetric heat source. As [111] describes, the finite volume method can be used to solve this equation by partitioning the domain into numerous discrete elements, and transforming it into a set of discretized equations:

$$-2(K_x + K_y + K_z)T_p + K_x(T_c + T_w) + K_y(T_n + T_s) + K_z(T_t + T_b) + q_{vol}\nabla x \nabla y \nabla z = 0 \tag{2.2}$$

where K_x, K_y, and, K_z are the thermal conductances that element p shares with its neighbors in the x, y, and, z directions and ∇x, ∇y, and ∇z are the element sizes in the x, y, and z directions, and the subscripts e, w, n, s, t, and b refer to the east, west, north, south, top and bottom directions relative to element p.

The Fourier model is capable of accurately modeling the thermal effects only at feature length scales much longer than the mean free path of phonons [103]. As a result, it can be used to model the thermal effects from the chippackage level down to the functional unit level. To perform thermal analysis at device level, the computationally-expensive BTE model should be utilized [59].

2.2 A Survey on Available Academic Thermal Models

HotSpot [61] is a fully parameterized, boundary condition independent, compact thermal model which can be used for preliminary exploration of thermal characteristics of electronic components at design time as well as when they exist as real hardware. HotSpot provides detailed temperature distribution at different levels such as silicon die layer, package and heat sink surface. It is capable of estimating steady state as well as transient temperatures in two different (faster) block and (more accurate) grid modes. The easy-to-use HotSpot user interface allows designers to perform complete simulations in a short period of time.

Sridhar et. al. introduced 3D-ICE [120], a thermal modeling tool which is specialized to support 3D-ICs. It also takes on-chip inter-tier liquid cooling into account for thermal simulation.

Allec et. al. introduced ThermalScope [17]. The work was then followed and extended by Yang et. al. to create ISAC [141] and also by Hassan et. al. which resulted in creation of NanoHeat [58]. These packages build a platform capable of performing thermal simulation at different scales of spatial and temporal resolutions. In [58] the granularity of the defined mesh to solve the heat transfer equation gets adjusted dynamically so that it uses higher spatial resolution at coordinates with higher temperature gradient. The tool provides two solvers, one based on Fourier heat conduction, and the other based on Boltzman heat transfer equations (BTE)[117]. The tool is claimed to be able to estimate temperature variations at transistor device resolution which are missed by traditional methods.

Chandra et. al. [87] described an efficient multi-griding approach for thermal simulation of ICs. The method also supports 3D structures.

Tang et. al. [128] introduced an improved adaptive finite element method for detection of hotspots in the chip as well as the PCB. In these papers, the resolution of the defined mesh which is used to solve the heat equation is refined adaptively according to the physical and geometrical characteristics of the chip in each region. This results

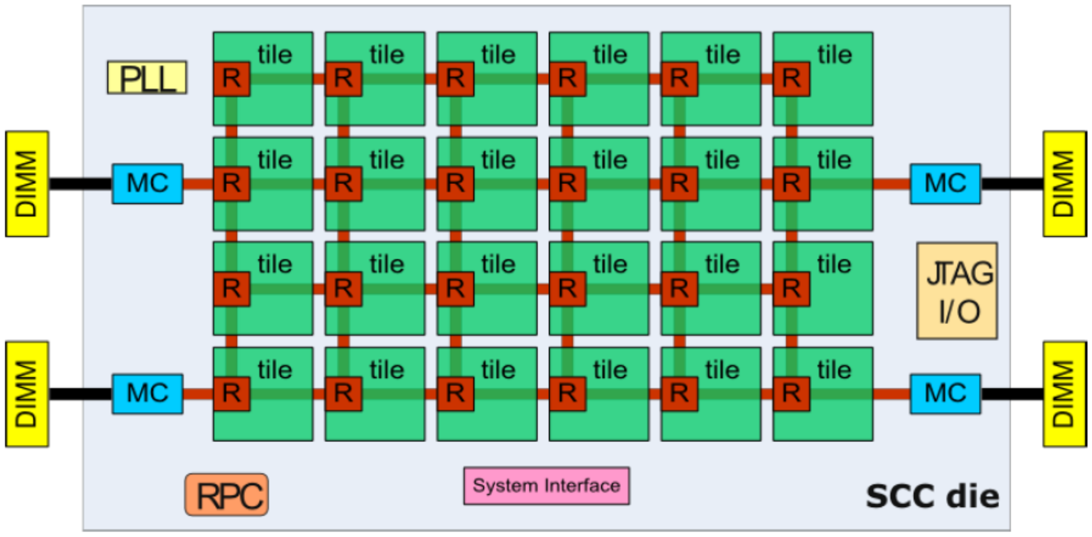


Figure 2.1: Block diagram of of Intel SCC.

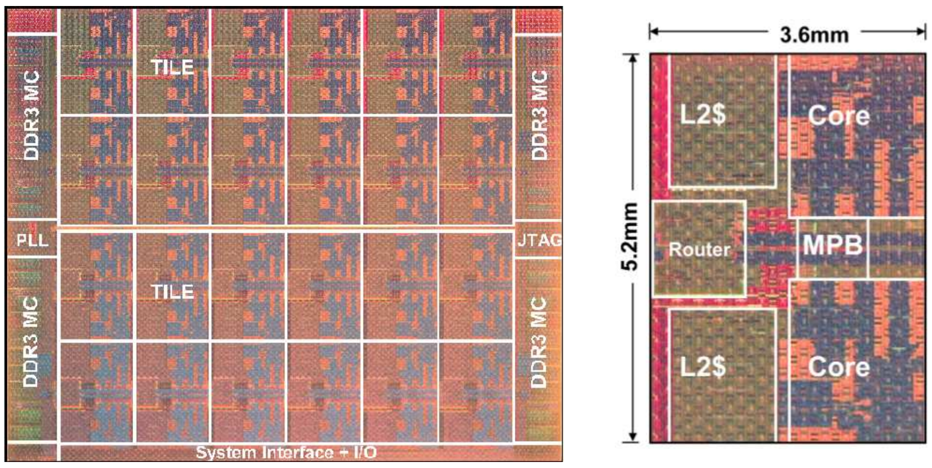
in improved simulation speed while keeping accuracy at acceptable level.

Fourmigue et. al. [47] introduce ICTherm, a tool developed for thermal analysis of 3D chips. The tool first performs a thermal evaluation of the target with a very high spatial resolution. Based on the results it creates a multi-granularity mesh which is used for thermal simulation.

2.3 Intel Single Chip Cloud Computer

The Single-Chip Cloud Computer (SCC) experimental processor [60] is a 48-core concept vehicle created by Intel Labs as a platform for many-core research. It has 24 dual-core tiles arranged in a 6x4 mesh. Each core is a P54C core. Each tile can be configured to run at its own clock frequency independent of other tiles. There exist 8 voltage islands on SCC die. The SCC die has four on-die DRAM memory controllers. Each DRAM controller is responsible for handling memory transactions of 12 tiles of the chip. Figure 2.1 shows a block diagram of the SCC chip. A Network-on-Chip (NoC) connects all of the on-die units together. SCC hardware is connected to its host PC through *System Interface* operating over a PCI Express link.

SCC also integrates a small amount of fast local memory located in each tile.



(a) SCC die photo (b) One Tile

Figure 2.2: Die photo of of Intel SCC.

Message-passing support is provided and use shared regions of local memory or off-die main memory.

Each tile integrates two thermal sensors based on a couple of ring oscillator, one positioned in proximity of the router and the other positioned close to the top core L1 cache. The SCC hardware includes power sensors capable of measuring the full SCC chip power consumption. Figure 2.2 shows the die photo of Intel SCC as well as the dimensions of one tile.

We develop an infrastructure to accurately track performance, power, and temperature of the SCC at runtime with very low performance overhead. Our setup collects performance counter data from each core, tracks main memory accesses, logs messages passed among cores, and measures power and temperature [2].

2.4 Thermal Simulation for Intel SCC

To required inputs to perform temperature estimations of a device are the followings: 1- The thermal floorplan, indicating the coordinates and dimension of each on die unit. 2- The estimated power values for each unit. These numbers are calculated by the developed power model. 3- The chip/package physical characteristics, which are provided by the manufacturer or measured using the real hardware.

The thermal floorplan of SCC has been derived directly from SCC specifications [60].

Table 2.1: Power break-down of different units in Intel SCC for two different extreme cases.

Unit Name	Full Power Cores 1 GHz, Mesh 2 GHz	Low Power Cores 125 MHz, Mesh 250 MHz
CPU Cores	87.7	5.1
DRAMs	23.6	17.2
Clocking	1.9	1.2
NOC (Routers)	12.1	1.2
SIF	≈ 0	≈ 0
Total	125.3	24.7

The input power traces of each block composing the floorplan has been partially extracted from the real power measurements by interpolating the power brake-down in [60]. For CPU cores, we develop a standalone power model. To assess model accuracy, we perform a comparison with real measurements under various workloads.

2.4.1 A Power Model for Intel SCC

Accurate thermal simulation requires detailed power traces for each floorplan block. As we can see from the floorplan there are the following basic blocks:

- CPU cores.
- DRAM controllers.
- Router + Clock.
- Serial interface engine (SIF) I/O.

We can obtain the power break-down for these blocks from [60]. In table 2.1, we show the values for two extreme stress workloads.

From Table 2.1 we see that the main contribution to the total power is given by the CPU and the DRAM controllers. Unfortunately, the power probes available on SCC are not sufficient to probe directly the power consumption of each functional unit. Indeed the available probes are:

Table 2.2: Workloads used for stressing SCC. (One cache line of SCC is 32 bytes).

Workload	Description
L1	16 KB circular buffer size. Data increment of 32 Bytes (1 cache line).
L2	32 KB circular buffer size. Address increment of 32 Bytes (1 cache line).
L2-2Access	32 KB circular buffer size. Address increment of 16 Bytes (2 access for cache line).
DRAM	4 MB circular buffer size. Address increment of 32 Bytes (1 cache line).
DRAM-2Access	4M B circular buffer size. Address increment of 16 Bytes (2 access for cache line).

- Analog portion of DRAM memory controller.
- Digital portion of DRAM memory controller and SIF.
- CPU cores, Routers and global clocking.

We first evaluate the sensibility of these probes to different levels of utilization of SCC. We then create a power model suitable to estimate the power consumption of each CPU core given its utilization and workload properties. This will allow us to feed each functional unit in the thermal floorplan with the proper power traces. The different stress patterns are obtained by executing different sets of synthetic benchmarks. Each synthetic benchmark is made-up of an infinite loop where an ALU operation is executed on a circular data buffer. The dimension of the circular buffer increases within different benchmarks: moving from 16 KB to 4 MB. At each iteration it is executing a read-write to an entry of circular buffer that moves with an incremental step of one cache-line. By doing that, the various synthetic benchmarks are capable of hitting always a data in the L1 cache, missing the L1 cache but hitting the L2 cache and missing L1, L2 and hitting the DRAM. These generate different memory stress patterns. Table 2.2 describes workloads properties of different synthetic benchmarks.

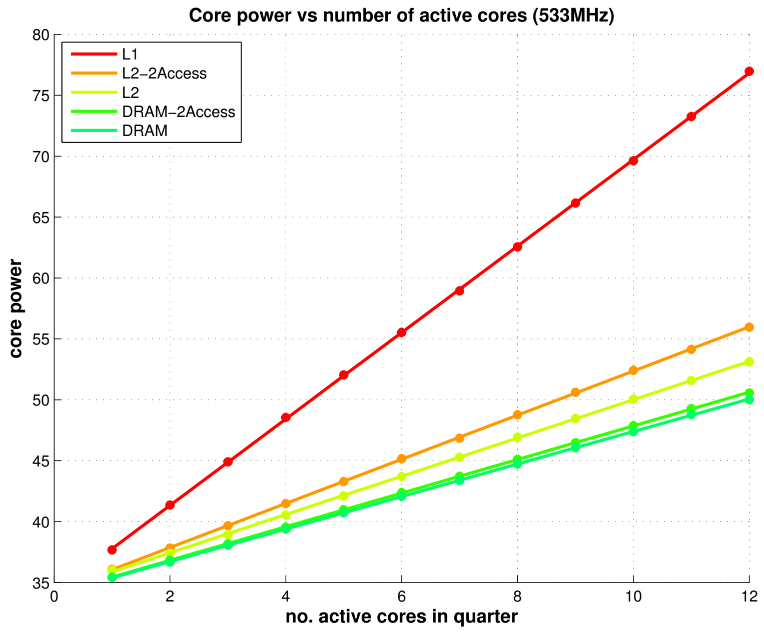


Figure 2.3: Core power consumption with number of active cores in each cluster for each of the workloads. Dots are measured data; Lines are the result of linear fitting.

2.4.1.1 SCC Power Measurement

The first analysis we performed has highlights how the core power changes in between different cores and workload under stable conditions. To do that, we divided SCC into 4 quarters of 12 cores each. Then for each of them we increase the number of active cores. We repeat this test for different synthetic benchmarks. As can be seen in Figure 2.3, the power increases linearly with the number of active cores for all of the stress workloads. This suggests that the contribution of the core power to the total chip power consumption is independent of the core position. Thus for the workload used, superposition principle is valid and we can obtain the single core power by dividing the total power to the number of cores.

The second test aims to highlight the effect of frequency scaling on different benchmarks. Figure 2.4 shows the cores power consumption when all of them are executing the same benchmark while scaling the tile frequencies. We can notice that the power changes linearly with the frequency with different slopes for each benchmark. Intuitively this is due to the different CPU-usage of the different synthetic benchmark. A common metric of CPU-load [39] is the Clock Per Instruction (CPI). This can be

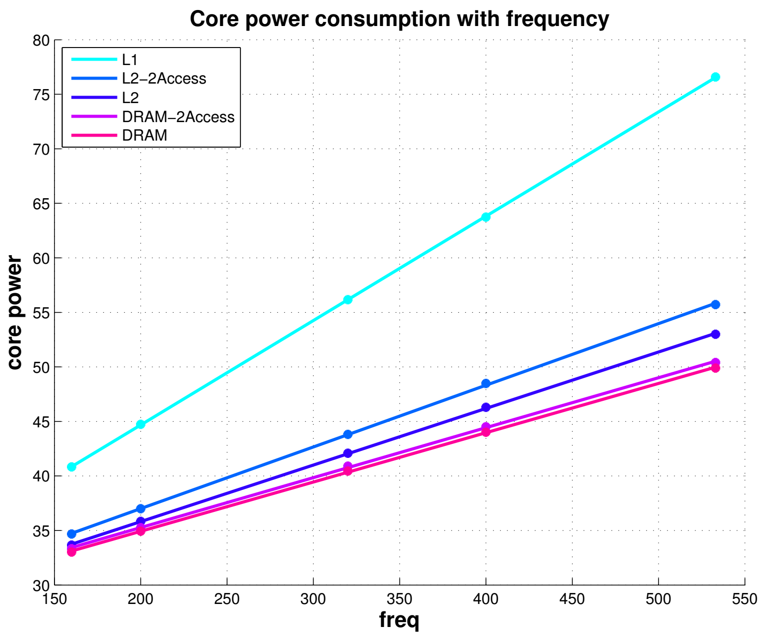


Figure 2.4: Core power consumption for each workload for different frequency values. Dots are measured data values. Lines are the result of linear fitting.

directly measured at run-time by using the performance counters.

Figure 2.5 shows the CPI of the different benchmarks while changing the tile frequency. We can notice that for L2 and L1 benchmarks the CPI does not change with the frequency. This suggests that both the L1 and L2 are in the same clock domain of the tile. For the DRAM benchmark instead the CPI scales with the frequency and the slope is not constant but changes itself with the frequency.

These effects can be explained by looking at the digital part of DRAM controller power consumption as shown in Figure 2.6. We notice the same effect. Indeed the power of the DRAM controller shows a saturation effect at higher frequency. This can be explained by bandwidth saturation due to the higher memory accesses issued at higher frequencies.

2.4.1.2 Power Model for SCC CPU Cores

We combine the core power data values, to derive a power model suitable to predict the core power consumption under different workload and tile frequency. For each core the power consumption can be split in two main contributions: the idle power and

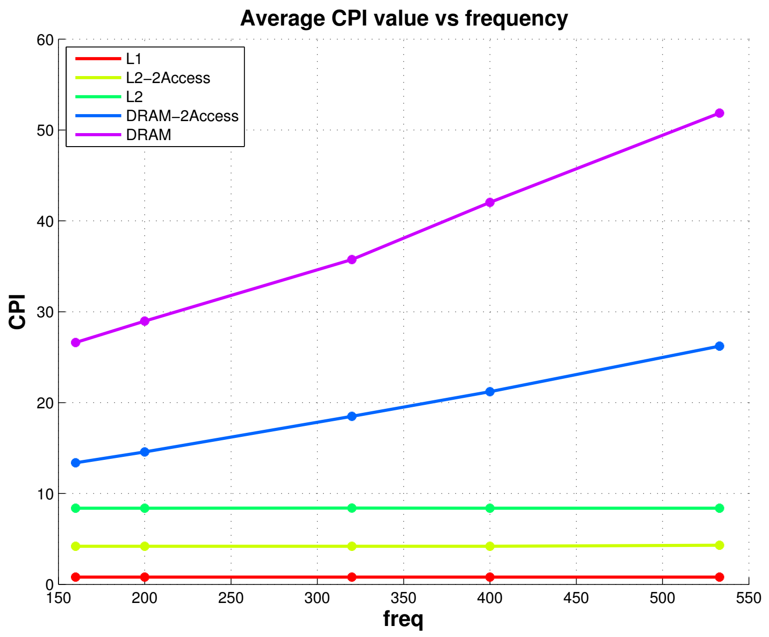


Figure 2.5: Changes of CPI for different values of frequency and different workloads.

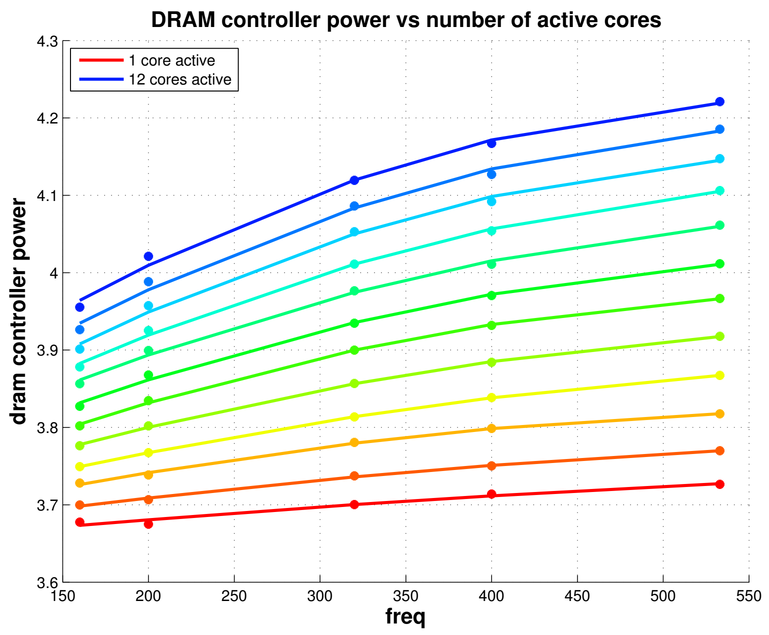


Figure 2.6: Power consumption of DRAM controllers as a function of frequency and number of active cores running.

the active power ($P_{core} = P_{active} + P_{idle}$). Whereas P_{idle} can be modeled as a function of only the frequency ($P_{idle} = g(f_{core})$), the active one can be modeled as a function of CPI and frequency ($P_{active} = g(CPI_{core}, f_{core})$). As a result, idle power can be effectively modeled as linear regression of the core clock frequency ($P_{idle} = b + a.f_{core}$). Using the experimental data a values $a = 0.35 \times 10^{-3}$ and $b = 0.3872$ are obtained. By using equation 2.3 as the fitting function, a closed form model can be generated for the per-core active power.

$$g(CPI, f) = (a + b.CPI^{c}).f + d + (a' + b'.CPI^{c'}).f \tag{2.3}$$

In equation 2.3, $g(CPI, f)$ is the per-core active power value. We use a least square optimization algorithm to find the optimal coefficients that minimizes the error value between estimated and real data. Values of $a = 0.0131$, $b = -0.240$, $c = 0.085$, $d = 0.021$, $a' = 0.0131$, $b' = 0.215$ and $c' = 0.02$ are obtained for the coefficients of equation 2.3 as result. Figure 2.7 shows the fitting performance. Since no performance counters are available in SCC to probe on-line the DRAM usage, we decided to use directly the power measurement of the dram controller as input to the thermal model. We feed each memory controller floorplan block in the thermal simulation with $1/4$ of the measured DRAM controller power consumption.

2.4.2 SCC Thermal Sensors

SCC integrates in each tile two built-in thermal sensors. The first thermal sensor is placed close to the router and the second one is placed near the L1 cache of the bottom core. Each thermal sensor is composed of two ring oscillators and the sensor output (TS) is the difference of the two oscillators clock counts over a specific time window t_W. The difference is proportional to the local die temperature (T) [74]. For each tile, the time window (t_W) can be programmed through a per-tile control register [73] as a number of tile clock cycle (NCC): $t_W = NCC/f_{Tile}$. Thus it needs to be updated each time the tile frequency changes. There is a linear relationship between the value of TS and the temperature (T).

$$TS = (A + B.T).t_W \tag{2.4}$$

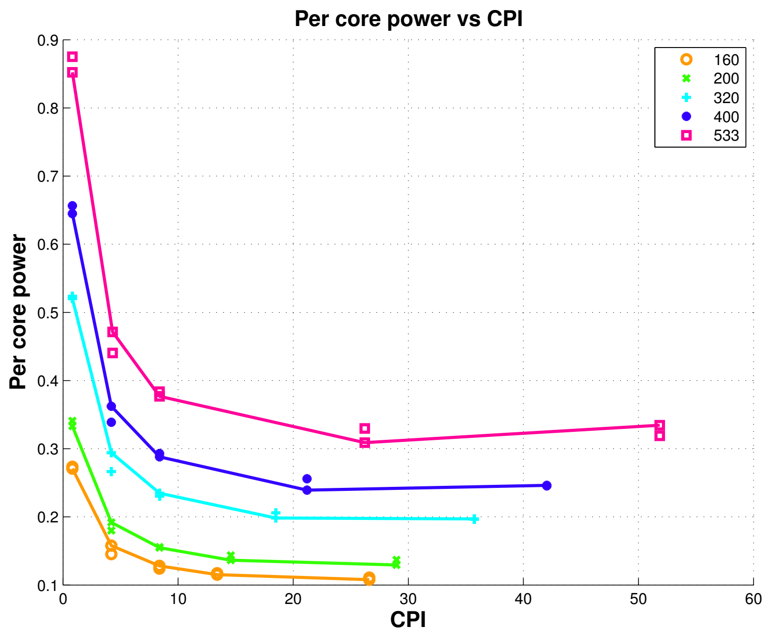


Figure 2.7: Power consumption of one CPU core based on CPI values and running frequency. Dots are measured values. Lines are the model output.

In Equation 2.4, $B < 0$ meaning that TS decreases with the rising of temperature and $A > 0$. TS is always positive and in the thousands. The A and B coefficients are completely different for each of the on-chip sensors.

We first evaluate the spatial variation of the thermal sensor values under two different homogeneous stress conditions. Figure 2.8 shows the results of this test. The X axis report the thermal sensors readings moving from the bottom-left corner to the top-right one of SCC (odd ones refer to router sensors whereas even ones to core sensor). The dashed line shows the thermal sensors output when all the cores are in idle state (no task allocated) and are running at the smallest possible clock frequency (100 MHz): the coldest operating condition of the chip. Instead, the solid line shows the thermal sensors output when all the cores are executing a power virus while running at higher frequency (533 MHz).

In the coldest point we can assume that the real silicon temperature is roughly constant across the chip area. However we can notice a strong variation in the raw sensor values (> 50%). Moreover, it is notable that idle and full-load plots are very similar in shape but for the hotter case (all cores busy) all the sensors output are significantly

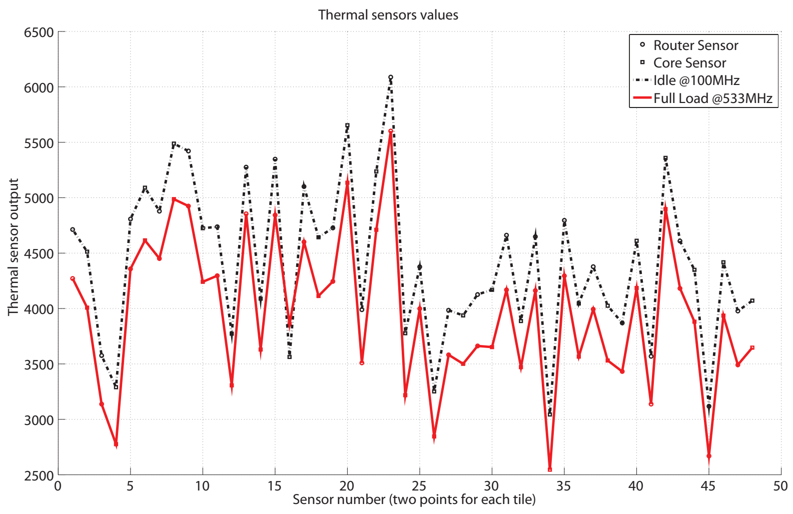


Figure 2.8: Raw thermal sensor values for two different tests.

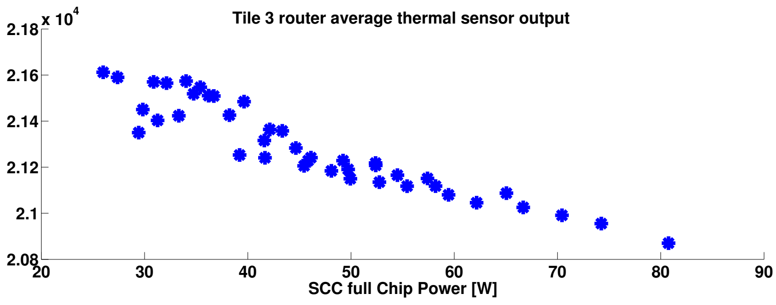


Figure 2.9: SCC thermal sensor readings with the change in total chip power consumption.

lower, as expected because sensor count decreases when temperature is high. However it must be noted that the variations due to temperature differences between minimum and maximum load conditions is less than 20% compared to the un-calibrated spatial sensors variation. This clearly highlights the strong need to the thermal sensor calibration step. We perform a second test with the goal of evaluating the relation between SCC power and thermal sensor output while executing different benchmarks and while running all the tiles together at different frequency levels. Figure 2.9 shows the results. We can recognize that sensor values are linear with the power consumption. This property can be exploited to characterize the thermal sensors taking advantage of the linear relation between temperature and power (in steady-state condition).

2.4.3 Thermal Sensors Calibration

If we consider the thermal transients are expired and we are in steady-state and we have a stable workload homogeneously distributed along the multi-core surface by executing same load/task in all of the cores, from equation 2.4 we can write the thermal sensor output as a direct function of full chip power consumption and ambient temperature (which can be measured easily by the end-user).

$$TS_i = A_i + B_i.T_i = A_i + B_i.(K_i.P_{core} + T_{amb}) = A_i + B_i.K_i.P_{core} + B_i.T_{amb} \tag{2.5}$$

Equation 2.5 can be formulated for each thermal sensor (i) as a least square problem where the unknown parameters are A_i, B_i and $B_i.K_i$, and the input data are TS_i, P_{core} and T_{amb}. Now if for each sensor i, we generate a cloud of N tuples $\{TS, P_{total}, T_{amb}\}$ by stressing all the cores of the target multi-core together at different frequencies and workloads we can calculate the desired coefficients by solving the obtained system of equations [1].

$$\begin{cases} Y_i = \{TS_{i,h}\}, h = 1, \cdots, N \\ X_i = \{1, P_{core,h}, T_{amb,h}\}, h = 1, \cdots, N \\ \Theta_i = \{\alpha_i, \beta_i, \gamma_i\}, \alpha_i = A_i, \beta_i = B_i.K_i, \gamma = B_i \\ Y_i = X_i.\Theta_i \Rightarrow \Theta_i = X_i^{\dagger}.Y_i \end{cases} \tag{2.6}$$

2.4.4 Hotspot Thermal Model for Intel SCC

We have developed a thermal model for SCC using HotSpot [61] thermal simulator. This model is capable of predicting SCC chip temperature values at different on-die locations according to input power to the chip. Since there is no public available information on SCC physical process parameters and chip package characteristics, we use the typical ones as provided in the default HotSpot model. Hotspot is capable of modeling package heat-sink and forced air convection (using fan). In this case, package convection resistance is computed using the details provided in package model file by the user. This file contains parameters related to dimension of the heat-sink, fan and also rotation speed for the fan. We obtain these by performing measurements on the

Table 2.3: Important Hotspot thermal model configuration parameter values for Intel SCC.

Parameter	Value
Chip Thickness	0.15 mm
Silicon thermal conductivity	100 $W/(m.K)$
Silicon specific heat	1.75 $J/(m^3.K)$
Heat spreader side	0.03 m
Heat spreader thickness	1 mm
Heat spreader thermal conductivity	400 $W.(m.K)$
Heat spreader specific heat	3.55 $J.(m^3.K)$
Interface material thickness	2.0e-2 mm
Interface material thermal conductivity	4.0 $W/(m.K)$
Interface material specific heat	4.0e6 $J/(m^3.K)$
Heat-sink fin height	8 cm
Heat-sink fin width	1 mm
Fan radius	9 cm
Fan motor radius	3 cm
Fan rotation speed	5000 rpm

real SCC heat-sink: ($H = 8$ cm, $W = 9$ cm, $L = 9$ cm). For the rotation speed of the fan on SCC platform, there is no direct measurement capability. As a consequence we used a typical fan rotation speed for high performance CPU cooling systems (5000rpm). Table 2.3 shows values of important parameters used for Hotspot thermal model.

2.4.5 Validation of Thermal Model

In order to evaluate the performance of our thermal model, we create a set of test cases. Each test case is composed by:

- A randomly created vector of active cores. This vector indicates for each core, if the core will execute a workload or will be idle.
- A randomly created vector of workloads that will be executed on SCC cores. Each element in the vector specifies one of the available stress workloads. (L1, L2, L2half, DRAM, DRAM-half)

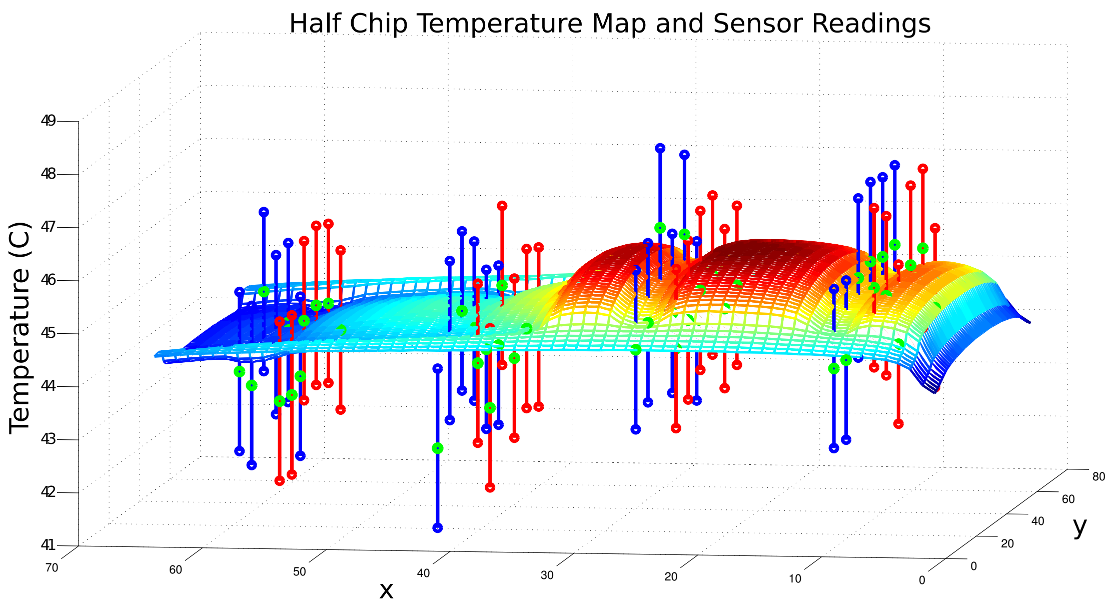


Figure 2.10: Comparison of measured temperature values with Hotspot ones when half of the chip is at idle state and the other half is at full load.

- A randomly created vector of tile frequencies. The items of this vector specify the frequency (between 533 MHz to 100 MHz) of each SCC tile.

We apply each test case to real SCC platform and to our power and thermal model. We then collect the results of each case and compare together. Table 2.4 shows the results. The bitmap in the table shows the workload executed on each core and the tile frequency used for each of the cores. The bitmap consists of two rows. Each row contains 48 rectangles. In the top row each rectangle with gray scale, indicates the workload executed on the specific core. The second row instead shows the value of tile frequency. Again the darker the rectangle shows higher frequency.

As we can see in Table 2.4, our power model is capable of estimating SCC core power consumption with a very good accuracy. Even if the max error value in temperature is 2 °C, the average difference between Hotspot estimation and measured values by thermal sensors is less than 1 °C.

Figure 2.10 shows the SCC thermal map obtained from calibrated sensors along with the hotspot simulation when half of the chip is idle and the other half is working at full load. The surface is the output temperature of the thermal model and bars

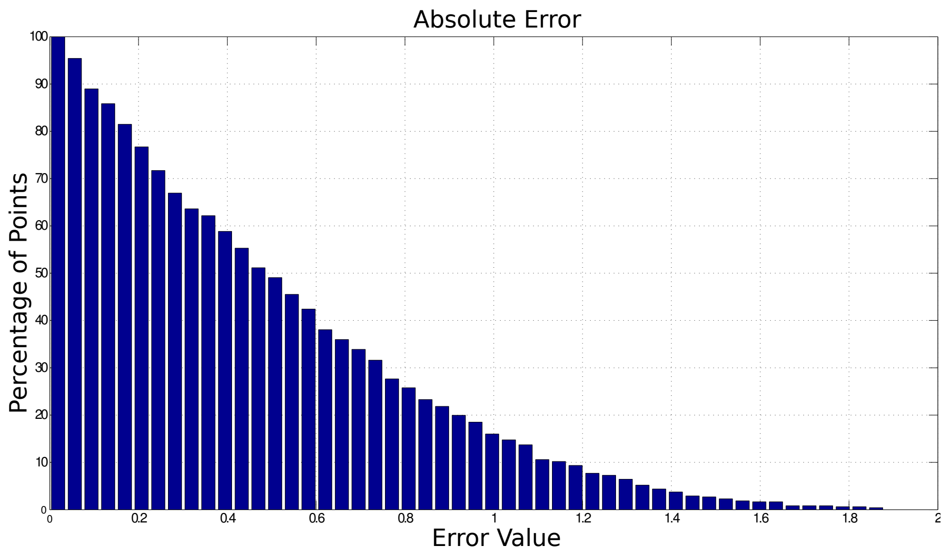


Figure 2.11: Statistical distribution of error values. For each error value on x axis, the value on y axis shows the percentage of points with error value higher than the specified error value.

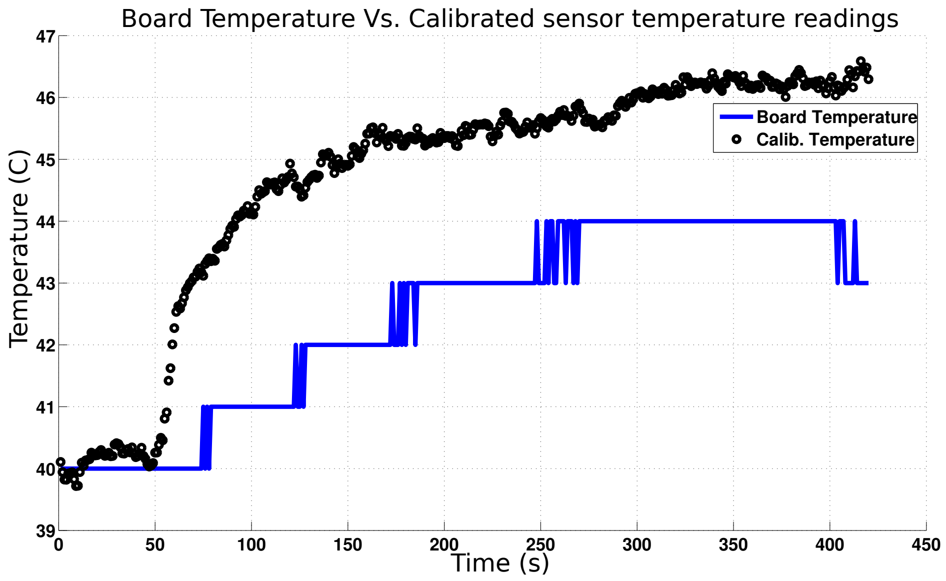


Figure 2.12: Temperature readings by the thermal sensor located on the main-board of SCC and one of on-die thermal sensors.

Table 2.4: Comparison of real measurements with model computed data for 10 diifferent tests. Ambient temperature $T_{amb} = 40$ °C.

Test No	Measured Core P. (W)	Estimated Core P. (W)	Temperature Mean Square Error (°C)	Temperature Max Error (°C)
1	41.23	40.91	0.59	2.22
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2	41.79	41.57	0.31	1.10
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
3	40.68	40.38	0.46	1.84
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
4	41.73	41.41	0.53	1.49
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
5	40.05	39.70	0.55	1.88
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
6	40.57	40.28	0.61	2.03
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
7	40.71	40.38	0.55	1.74
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
8	42.06	41.76	0.47	1.79
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
9	42.12	41.85	0.44	1.69
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
10	40.89	40.78	0.60	1.73
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

represent the calibrated sensors outputs with a tolerance range (1.5 °C). As we can see, our thermal model is capable of tracking real temperature values under a random workload with good accuracy.

Figure 2.11 shows the statistical distribution of error between real results and hotspot one. As can be seen, less than 10% of points have error values larger than 1 °C. A comparison between the temperature measured by the thermal sensor located near the SCC chip on the main-board and the calibrated sensor readings of SCC for a sample stress workload is shown in Figure 2.12. As we see, the temperature measured by on-die thermal sensors is always higher than on-board sensor.

2.5 Quantifying the impact of frequency scaling on energy efficiency of SCC

The goal of this Section is to analyze the impact of core frequency perturbations on the performance of many-core systems with Message Passing Interface (MPI). We use the Intel SCC as our test bed which incorporates features such as a network-on-chip (NoC), DVFS capabilities, and support for MPI.

To analyze the impact of frequency changes on the performance for many-core systems with MPI, we develop a benchmark suite for the SCC. The benchmarks in the suite cover a wide range of workload scenarios such as applications with different levels of communication distances and intensity or applications that stress different levels of the memory hierarchy.

We conduct a large set of experiments using the monitoring infrastructure and the benchmark suite to measure the impact of frequency changes on performance and power.

2.5.1 Tailored Benchmarks

We utilize a set of benchmarks to assess the performance of the SCC under a variety of operating conditions. In addition to expanding the benchmarks provided by Intel, we design several micro-benchmarks to stress different parts of the system. We select the following application and synthetic benchmarks as they provide a heterogeneous set of performance data to use during analysis and creation of our model.
Intel benchmarks:

- *Share*: Tests the off-chip shared memory access.
- *Shift*: Passes messages around a logical ring of cores.
- *Stencil*: Solves a simple PDE with a basic stencil code.
- *Pingpong*: Bounces messages between a pair of cores.
- *NPB*: NAS Parallel Benchmarks, LU and BT.

Custom-designed micro-benchmarks:

Table 2.5: Benchmark Categorization

Benchmark	L1CM	Time	Msgs	IPC
Share	High	High	Low	Low
Shift	High	Low	High	Medium
Stencil	Low	Low	Low	High
Pingpong	High	Medium	Medium	Low

- *Bcast*: Broadcasts messages from one core to all other cores.
- *DRAM*: Executes an ALU operation on a circular buffer in memory. At each iteration a read-write is performed for one entry of the circular buffer. The dimension of circular buffer is 4MB.
- *L1 and L2*: Have the same principle as the DRAM benchmark. Circular buffer size for L1 is 16KB and buffer size for L2 is 32KB.

Table 2.5 categorizes the Intel benchmarks based on IPC, L1 instruction misses (L1CM), number of messages (Msgs), and execution time. All parameters are normalized with respect to the number of instructions executed to enable a fair comparison. Each benchmark in this categorization runs on two neighbor cores on the SCC (only 2 cores active). We observe that Share does not have messages and is an example of a memory-bounded application. Shift represents a message intensive application and Stencil represents a high-IPC application. Finally, Pingpong is a low-IPC application with a large number of L1 cache misses. Note that Stencil, Shift, Share, and Pingpong benchmarks all rely on the blocking send / receive calls from the RCCE API [134].

We designed the Broadcast (Bcast) benchmark based on Pingpong, which sends messages among cores and test latencies. Instead of having a source and a single destination as in Pingpong, Bcast sends messages from a single core to multiple cores. DRAM, L1 and L2 benchmarks are custom-designed applications that do not use the RCCE library. We use these memory benchmarks to compare the execution time of memory-intensive (DRAM) or cache-intensive (L1 and L2) applications in our analysis and to analyze the tradeoffs under frequency scaling. The benchmarks are run with the following configurations to cover a wide range of workload scenarios.
Intel benchmarks: Stencil, Shift, Share, and Pingpong benchmarks run on *pairs* of cores in the following configurations:

- *0-hop*: Cores on the same tile. (e.g., cores 0-1)
- *1-hop*: Cores on neighboring tiles (e.g., cores 0-2)
- *2-hops*: Cores on tiles that are 2-hops distance away (e.g., cores 0-4)
- *3-hops*: Cores on tiles that are 3-hops distance away (e.g., cores 0-6)
- *8-hops*: Cores on corners (e.g., cores 0-47)

We run either a single pair (2 cores active) or concurrently run 24 pairs for each benchmark (48 cores active) in the experiments with Intel benchmarks.
NPB Benchmark: The LU benchmark runs on 32 cores and the BT benchmarks runs on 36 cores. These choices are due to restrictions with LU and BT software preventing execution on all 48 cores.
Bcast Benchmark: The Bcast benchmark is run with 1 core sending messages to N cores ($1 \leq N \leq 47$).
DRAM, L1, L2 Benchmarks: Our custom-designed memory benchmarks are single-threaded benchmarks. We run 48 instances on 48 cores.

2.5.2 Experimental Evaluation

We carry out a test to highlight how different benchmarks behave under frequency perturbation. For this experiment, we execute each of the Intel benchmarks on two cores of the SCC. One of the cores ($core_A$) is always Core0 (corner core) while the second one ($core_B$) moves step by step towards the opposite corner from 1-hop distance to 8-hop distance in the SCC floorplan. Then for each of these configurations we perturb frequency of the tiles of the running cores to generate the following frequency patterns: $\{tile_A, tile_B\}$: $\{f_{min}, f_{min}\}$, $\{f_{max}, f_{min}\}$,$\{f_{min}, f_{max}\}$,$\{f_{max}, f_{max}\}$. In our experiments, f_{max} is 533 MHz and f_{min} is 166 MHz. These choices for f_{max} and f_{min} were made considering the stability of the SCC system.

During each run, we probe the (1) execution time overhead, (2) the full chip power saving, (3) the energy saving, (4) instructions per second (IPS), (5) message density, and (6) memory access density. For the first three metrics, the baseline has the $\{f_{max}, f_{max}\}$ setting and $core_A$ is adjacent to $core_B$. The message density is computed as the number of messages sent and received by a given core divided by the total number of instructions, whereas the memory access density is computed as a ratio of the non-cacheable memory read performance counter over the total number of instructions[1].

In Figure 2.13 we show the results of the stress patterns for nearest and farthest position of $core_B$ (denoted with "near" and "far"). Bcast is an asymmetric benchmark, meaning the communication direction is always from a source core to a destination, and has a high message density. In contrast to the other benchmarks, the performance loss when only one core has lower frequency while running Bcast is significantly lower when $core_B$ (the destination core) is slowed down. This is not the case for the other benchmarks, as other benchmarks include bidirectional communication among cores. In addition, Bcast strongly benefits from running both cores at the same frequency, as the execution time overhead and the energy are lower compared to running cores at different frequencies. Note that as we do not scale the voltage of the cores, we do not observe energy savings when both the cores run at the minimum frequency.

Pingpong and Share show similar trends even though they are significantly different applications. Their execution times have lower sensitivity to frequency changes compared to other benchmarks. For Share, this effect can be explained looking at IPS, which is lower compared to the other benchmarks. Also, the memory read access statistics show that Share is memory-bound.

Shift has high message density and, similar to Bcast, its execution time strongly depends on the core frequency. Stencil, on the other hand, has low memory access density and high IPS. Stencil's throughput decreases significantly as we scale down the frequency of one core. In addition, similar to Share, Stencil's execution time increases when running on cores far from each other. This increase is mainly due to the usage of the shared memory buffers allocated off-chip (for Share) or in the MPB (for Stencil). For Stencil, increasing the distance reduces the throughput (IPS) considerably. The slow-down saturates when just one core runs at low frequency. In this case, scaling down the other core does not affect the execution time as Stencil uses barrier synchronization.

[1]Note that SCC does not include a performance counter to track the L2 miss rate. We tested the memory read performance counter with micro-benchmarks and verified that there is a strong correlation with off-chip memory access.

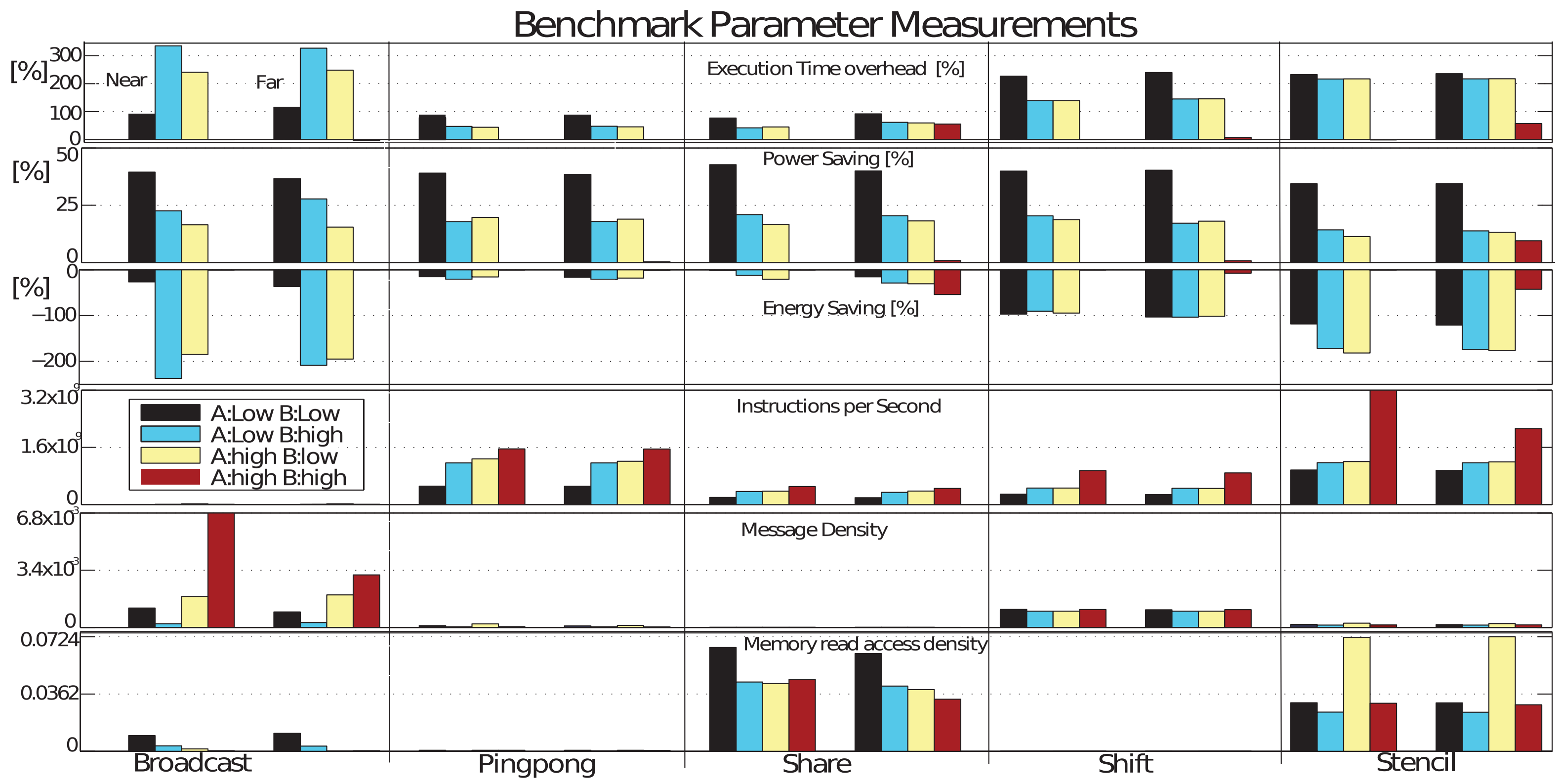


Figure 2.13: Sensitivity of the Intel SCC benchmarks to frequency scaling.

We observe that all the benchmarks benefit from having the core frequencies equalized. In fact, for most of the benchmarks we see significant energy savings when moving from only one core operating at low frequency to both cores operating at lower frequency. An unbalanced frequency configuration can lead up to 2x energy efficiency loss.

This analysis highlights the importance of predicting the impact of a generic frequency perturbation on the execution time of a parallel benchmark. In addition, it suggests that message density, IPS, and frequency can be utilized to estimate the changes in execution time.

Chapter 3

Temperature Variation Aware Delay, Power and Thermal Analysis

In this Chapter, we introduce MiMAPT (Micrel Multi-scale Analyzer for Power and Temperature) [5, 7], a completely stand-alone and fully functional software. It is capable of performing temperature variation aware delay/power and thermal analysis at RT and gate level with multiple scales of resolution and speed.

We first describe research works related to studying the effect of temperature variation on the design of ICs. Then we focus on the architecture and operation of MiMAPT.

3.1 The Importance of Temperature Variation Aware Analysis

The following research papers show the importance of accounting for temperature non-uniformities for accurate evaluation of digital designs.

Mcallister et. al. [138] consider the effect of non-uniform temperature distribution on the IR drop in 3D chips. It also accounts for the self-heating effect of power delivery network.

Li et. al. [88] study the effect of temperature variation on IR-drop and propose a solution for sophisticated creation of power-grid networks avoiding thermal-induced IR-drop hazards.

Haghdad et. al. [57] study the effect of temperature and supply voltage variations on the estimated power yield. They show a significant yield loss can happen, if the statistical measures of temperature and IR-drop are ignored.

Chakraborty et. al. [27] considers the effect of on-die temperature non-uniformities on the delay and skew of clock tree. It shows that clock trees designed using standard methodology with assumption of an uniform on-die temperature, may suffer from significant skew violations. To counteract this effect, it proposes new algorithms to optimize the clock tree accounting for temperature variations.

Wu et. al. [137] proposes a bus-driven floorplaning method for multi-core SoC designs considering thermal distribution of the chip.

Timar et. al. [131] introduced Logi-Therm. The tool performs concurrent electrical and thermal simulation of standard cell ASIC circuits. It takes standard cells of the digital design as basic building blocks and based on cell's power characteristics and switching activity calculates a power/thermal distribution map of the chip. Analysis done by Logi-Therm however are based on a fixed resolution.

Moreover, in [130] they consider the effect of temperature variation while doing timing analysis on a given design. The represented method to apply non-uniform temperature map to the design however, can not scale well with the size of the design. In fact, the proposed method updates the output of timing/power analysis tool based on a given temperature map, so that by re-analyzing the output one can have updated delay values. As a result,while the method by [130] can be used for small designs such as ring-oscillators, it will face difficulties handling large designs of several thousand cells.

In contrast, MiMAPT arms the timing/power analysis tool (e.g. PrimeTime [124]) to consider non-uniform temperature map of the die from the beginning of its analysis procedure. Thus the tool has already considered non-uniformities in the temperature when it generates the output reports. This is a significant advantage of MiMAPT over similar research works. Indeed, MiMAPT does not introduce any timing estimation system from its own, instead it is designed so that it integrates into standard chip design flow and enables the *industry trusted* timing analysis tool to produce more accurate timing estimates which also consider on-die temperature non-uniformities. As we show in Section 3.5, MiMAPT can be effectively applied to designs containing several IP blocks and millions of standard cells.

Yuan et. al. [54] propose a method of leakage reduction using high voltage threshold (HVT) and low voltage threshold (LVT) standard cells. They consider non-uniform temperature map of different die areas to estimate leakage and to determine if regular cells should be replaced with HVT cells to reduce leakage or with LVT cells to improve path delays. They create a simplified chip/package thermal model to estimate steady-state temperature map of the die, based on which decisions are made.

Meterelliyoz et. al. [100] study the key characteristics of cache memories under existence of hotspots. Toward this task, the paper focuses on basic building cells of the memory components, and represents their behavior with temperature. The paper uses available delay/power models of these cells to extract results. This work is only limited to cache memories however, MiMAPT temperature variation aware analysis can be used for any kind of integrated digital circuit. Moreover, MiMAPT utilizes standard cell libraries provided by silicon foundry and the standard Back-end flow power/timing analysis tools to study the effect of temperature variation on design characteristics.

Calimera et. al. [25] use inverse temperature dependence (ITD) effect of low power cells to perform dual threshold voltage synthesis of the circuits. Since HVT and LVT cell delays show contradictory trends with temperature for a specific range of supply voltages, sophisticated use of them during synthesis can result in circuits which have very low level of sensitivity to temperature changes. This work however, assumes a uniform temperature across entire die. By using MiMAPT, this work can be extended to also consider non-uniform temperature maps at the time of dual VTH cell assignment.

Considering commercial software packages, Gradient Design Automation is known to be able to perform concurrent power/thermal analysis with multiple scales of temporal and spatial resolution in transient and steady state [28, 29]. Compared to Gradient's solution, MiMAPT provides wider range of analysis speed and accuracy, since it basically performs thermal analysis at RT level, which is very fast, and switches to gate level only when higher levels of resolution are demanded. Moreover, in contrast to commercial solutions, MiMAPT is a free and open source tool. As we will show, it provides a research vehicle for accurate evaluation of temperature variation impact on deep sub-micron digital designs.

In fact, MiMAPT is an academic package which meets all of the following requirements together:

1. Power and temperature estimation at RT and gate level with different scales of

spatial resolution.

2. Seamless integration into major design tool flows and compatibility with widely used library standards.

3. Accounting for non-uniform on-die temperature distribution while estimating critical timing path delays, consumed design power and temperature map.

4. Compatibility and open interfaces with commercial and academic thermal analysis tools (such as Hotspot[61], 3D-ICE[120] and FloTHERM[98]).

5. Early multi-scale power and thermal analysis of a design while the building IP blocks are available as separate entities but the final chip is not implemented yet.

3.2 MiMAPT Architecture

MiMAPT mainly consists of two engines:

1. Temperature variation aware delay/power analyzer.

2. Multi-scale thermal simulator at RT and gate level.

In practice, these two engines operate jointly together to gain highest level of accuracy in results. We first describe how a given temperature map is applied to a design and how the effect of temperature variation is taken into account while doing delay/power estimations (Section 3.2.1). Then, we focus on algorithms related to multi-scale thermal simulation at RT and gate level and the joint operation of two engines (Section 3.2.2). Finally, we illustrate the capability of MiMAPT in merging several different IP blocks into a single virtual chip to perform analysis (Section 3.2.3).

3.2.1 Temperature Variation Aware Delay and Power Estimation

The algorithm described in this section is executed at gate level. It mainly provides the delay/power analysis tool with two sets of data for each of the standard cell instances in the design: 1) Operating temperature. 2) Temperature dependent delay/power cell parameters. The first item is obtained from the on-die temperature map. The second

Algorithm 1 Apply temperature map to design for power/timing analysis

```
Require: F = Thermal floorplan of the design
Require: T = Temperature map of the design
Require: C = Cell info (instance, location)
  for all $blocks\ b \in F$ do
      $< x_1, y_1, x_2, y_2 > \leftarrow boundary\ of\ b$
      $T_b \leftarrow Temperature\ of\ b$
      $A = \emptyset$
      for all $cells c \in C$ do
          $< l_x, l_y > \leftarrow location\ of\ c$
          if $(l_x \geq x_1), (l_x < x_2), (l_y \geq y_1), (l_y < y_2)$ then
              $c \in A$
          end if
      end for
      $cmd \leftarrow (A, T_b)$
      $Update\ cells\ operating\ condition\ (cmd)$
  end for
```

item is in fact the data provided by the standard cell libraries. The key point here is the correct selection of libraries which can provide sufficient data for an accurate delay/power estimation by the tool.

Algorithm 1 introduces non-uniform on-die temperature map to the delay/power analysis tool for a design. It receives a previously generated temperature map, and creates a set of required commands which will be processed by the timing/power analysis tool. These commands modify the operating condition of each cell based on the given temperature map. The inputs to the algorithm are: the temperature map and the created database which contains a list of design cell instance names, plus their physical on-die coordinates.

Our approach targets the effect of temperature variation only on standard cells of the design. For wires, as of our practical experiments, the effect of variation in temperature is small in comparison with standard cells. As a result, for routing resources we always use the typical operating corner. The modular architecture of MiMAPT however, considers the possibility of adding required extensions to take the effect of temperature variation on wire delays into account when they become strongly temperature sensitive.

Standard Cell Libraries: Algorithm 1 is assuming that the delay/power analysis software (e.g. Synopsys PrimeTime [124] or Cadence SoC Encounter [24]) is capable of considering non-equal temperature values for design cells.

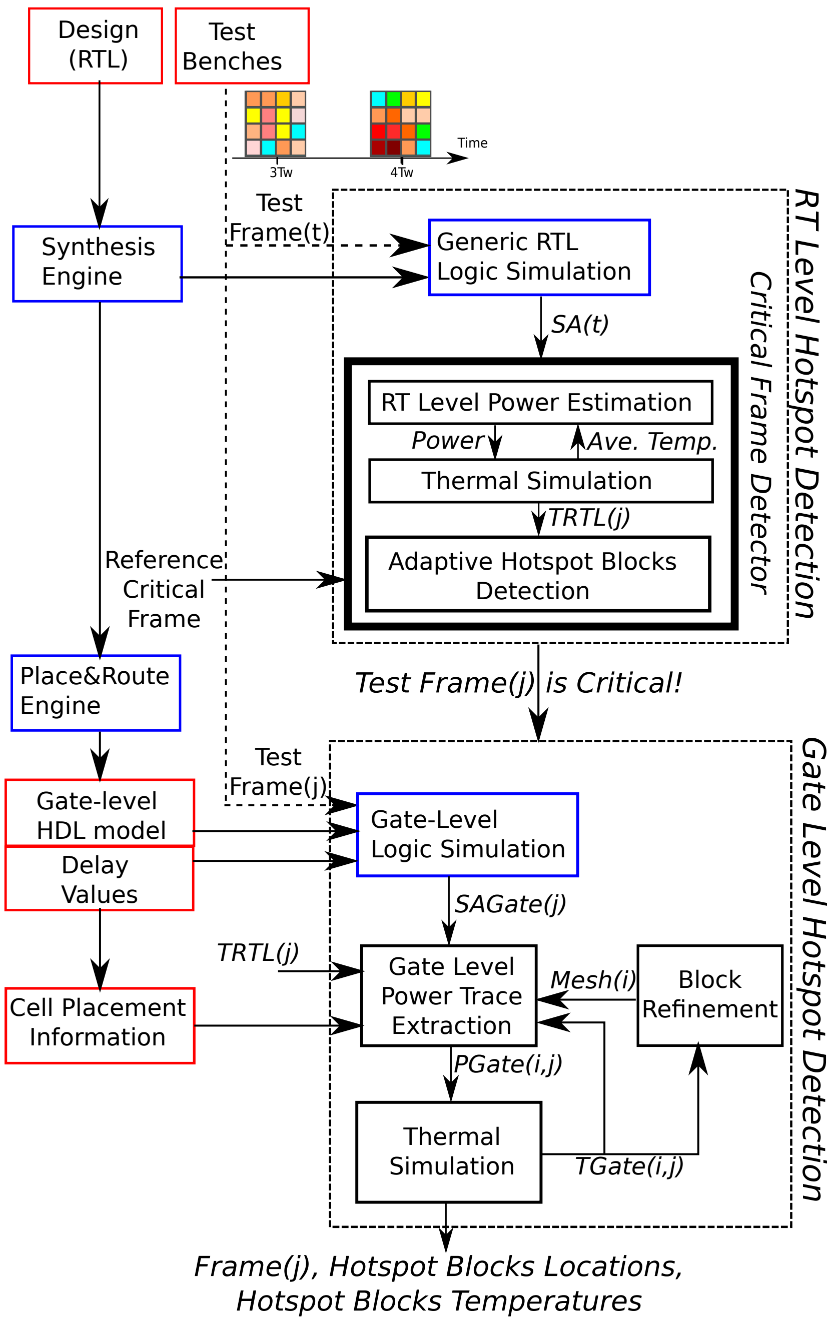
In contrast to traditional design flows which are based on Non-linear Delay Model (NLDM) cell libraries, MiMAPT requires the provided cell models to account for wide-ranging voltage and temperature variations. Examples include the gate model proposed in [63], Composite Current Source (CCS) [95] and Effective Current Source Models (ECSM) [52, 55, 105] This is due to the fact that, characterized libraries provided by silicon foundries usually describe the behavior of standard cells for limited number of operating corners. However, for temperature variation aware analysis, knowledge of cell behavior is required across a continuous range of temperature and voltages values. As a result, library characterization should be done for all of the possible temperature and supply voltage combinations (which is impossible) or, the data provided by characterized libraries at key corner cases should enable the tool to perform accurate enough corner case interpolation. This is what CCS and ECSM models provide.

3.2.2 Multi-scale Thermal Simulation at RT and Gate Level

We first describe the procedure for estimation of temperature at RT level and the adaptive method used for detection of thermal hazards at RTL. Then we focus on gate level thermal simulation where we represent the details on thermal simulation with multiple levels of spatial granularity.

Our approach leverages power analysis features provided by state-of-the-art commercial tools. Recent versions of logic synthesis tools (e.g. Cadence RC® and Synopsys DC®) are capable of estimating power at RTL based on switching activity obtained from functional logic simulation. As a result, power estimation at RTL can be done very fast, but it is not guaranteed to be fully accurate [23, 125]. On the other hand, very accurate power estimate at gate level can be obtained after (or during) back-end flow. The power analysis tool should be provided with the finished (placed, routed, clock tree synthesized) design and also switching activity statistics obtained from logic simulation of the finished gate level netlist with timing delays annotated through an SDF file.

In classical thermal simulation flow the power estimation at gate level is used to ob-



Design (RTL)
Test Benches
Time
3Tw
4Tw
Test Frame(t)
Synthesis Engine
Generic RTL Logic Simulation
SA(t)
RT Level Power Estimation
Power
Ave. Temp.
Thermal Simulation
TRTL(j)
Adaptive Hotspot Blocks Detection
Reference Critical Frame
Critical Frame Detector
RT Level Hotspot Detection
Place&Route Engine
Test Frame(j) is Critical!
Test Frame(j)
Gate-level HDL model
Delay Values
Gate-Level Logic Simulation
SAGate(j)
TRTL(j)
Cell Placement Information
Gate Level Power Trace Extraction
Mesh(i)
Block Refinement
PGate(i,j)
Thermal Simulation
TGate(i,j)
Gate Level Hotspot Detection
Frame(j), Hotspot Blocks Locations, Hotspot Blocks Temperatures

Figure 3.1: MiMAPT Block Diagram

tain per-cell power values. This fine-grain power map is then used as input to fine-grain thermal simulation [58, 61]. MiMAPT instead performs power/thermal simulation at RT level with the goal of avoiding time consuming gate level simulation when hotspots do not exist.

A die area is defined as hotspot when its temperature (T) is higher than a specified threshold (TH). Figure 3.1 shows the basic building elements of MiMAPT multi-scale engine which are divided into two major parts: RTL and gate level.

The simulation input sequence contains a set of subsequences called *Test Frames*. Basically, each *Test Frame* (*TF*) represents a different use case (stress workload) of the design and corresponds to a switching activity statistics for the circuit. For each *TF* thermal simulation will be executed at RTL (as described in Section 3.2.2.1). Examining the obtained temperature map, we dynamically switch to gate level if needed (Section 3.2.2.2).

3.2.2.1 RT Level Hotspot Detection

We perform logic simulation for each of the *TF*s at RTL to obtain switching activity ($SA(t)$). This information are then fed to the synthesis tool to obtain power estimation for each of the design sub-modules.

We define a new *thermal floorplan* for the design which divides chip area into equally sized rectangular blocks. For each floorplan block (FB), we calculate what percentage of each sub-module is located inside this block. Based on the percentage we add a fraction of sub-module's power to the total power of the block. Psudocode 2 shows this in more detail.

Figure 3.2 shows this procedure in more detail for a sample design. In this Figure, *(a)* shows the defined floorplan for the chip in the layout tool. *(b)* shows the traditional method of creating *thermal floorplan* in which we use the dimensions defined by the default chip floorplan directly. *(c)* shows the chip after placement, as we see the real placement of sub-modules is different than the default floorplan thus an accurate thermal simulation is not possible using traditional methods. In *(d)* we show our method of defining a new *thermal floorplan*. Each floorplan block contains one or more design sub-modules. The synthesis tool provides us with the power consumption of each sub-module. Using chip placement information that we have, we obtain the percentage of each sub-module inside each floorplan block by counting the number of cells owned by this sub-module inside the floorplan block. We suppose that the sub-module's power is uniformly divided between its cells thus, we use the obtained percentage to add the

Algorithm 2 Generation of power map for thermal simulation at RTL

```
Require: C = Cell placement info
Require: N = total number of floorplan blocks
Require: K = total number of sub-modules in design
Require: P = computed per sub-module power values
  for each block in floorplan do
      $blockPower \leftarrow 0$
      $A$ =(physical location of block ¡x1,y1,x2,y2¿)
      for $subModuleNo = 1 \rightarrow K$ do
          $t$ =(total number of sub-module leaf-cells)
          $c$ =(number of sub-module leaf-cells in $A$)
          $tc = t/c$
          $moduleBlockPower = P[subModuleNo] * tc$
          $blockPower \leftarrow blockPower + moduleBlockPower$
      end for
  end for
```

fraction of sub-module's power to the total power of the floorplan block.

Final obtained power map is then used by the thermal simulator [61] to estimate per-block temperature map. Thermal simulation is done in transient mode and its duration is equal to the duration of the *TF*. The output of each transient thermal simulation for each *TF* will be used as the initial temperature values for the thermal simulation of the next *TF*. Obtained thermal map for each *TF* is compared with a set of adaptive thresholds to identify if the *TF* contains critical areas. If this situation is detected we trigger the gate level hotspot detection for the same *TF*. As shown in experimental results, RTL hotspot detection executes significantly faster than the gate level simulation.

Estimated power at RT level is usually not equal to gate level power since the design is not fully synthesized yet. Consequently, using a unique temperature threshold value to identify hotspot blocks at RT and gate level may not lead to accurate detection of hotspots at RTL. Thus we use an adaptive method to detect hotspots at RTL. Psudocode 3 describes this in detail.

We first select the *TF* in which every design sub-module has highest level of activity and thus power, as reference. We perform thermal simulation for this *TF* at RTL and gate level and we mark critical floorplan blocks (FB) by comparing their temperature values at gate level ($CMapGate$) to a threshold ($THigh$) which contains a safe margin with respect to TH and is slightly lower (e.g. 1°C) than it. We use the computed critical

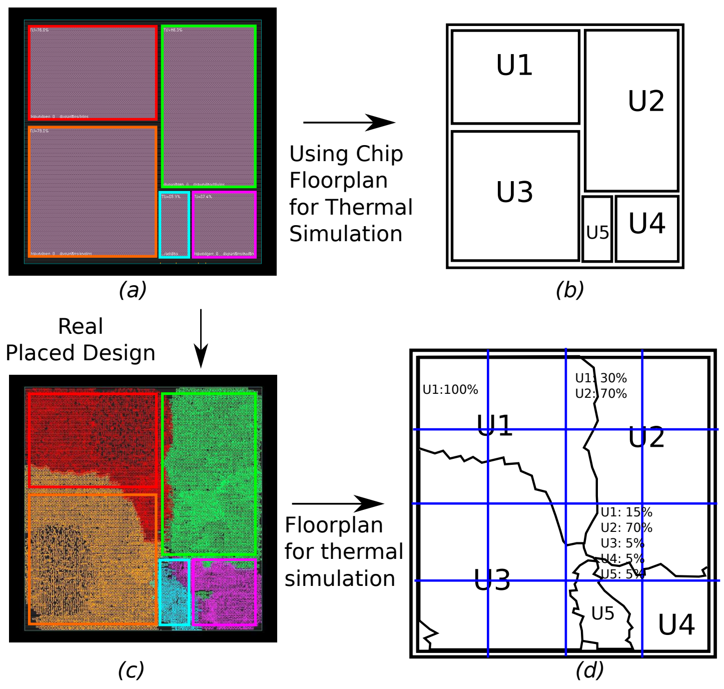


Figure 3.2: Generation of power map for thermal simulation at RTL

blocks map ($CritMatrix$) for identifying hotspots at RT level for the rest of TFs.

For each TF we perform thermal simulation at RTL and we compare each block temperature value with an adaptive threshold. If the block is marked as critical, we act more carefully, thus we create a reduced threshold value by multiplying a coefficient $A < 1.0$ to the reference block temperature ($CMapRTL$). If the block is not critical we use an increased threshold value by multiplying a coefficient $B > 1.0$ to the reference block temperature to avoid unnecessary hotspot detection. The smaller values for A make detection of hotspots at RT level safer however, they decrease overall operation speed.

3.2.2.2 Gate Level Hotspot Detection

If $TF(j)$ is detected as critical in RTL hotspot detector, it will be processed with high accuracy at gate level to correctly estimate the hotspot position and temperature. This is done by first performing logic simulation at gate level to obtain circuit switching activity ($SAGate(j)$) used for power estimation. The power map ($PGate(i, j)$) is then converted to temperature ($TGate(i, j)$) using an iterative multi-granularity meshing

Algorithm 3 Adaptive Hotspot Detection

```
Require: CMapRTL, CMapGate= Highest-workload frame, temperature map at RTL
    and gate-level
Require: TMapRTL= current frame temperature map
Require: THigh= threshold value for critical block
Require: A, B: Coefficients ($A < 1.0$) and ($B > 1.0$)
Require: N= Total number of thermal floorplan blocks
  for $i = 1 \to N$ do
      $t = CMAPGate(i)$
      if $t > (THigh)$ then
          $CritMatrix(i) = 1$
      else
          $CritMatrix(i) = 0$
      end if
  end for
  for $i = 1 \to N$ do
      $t = TMapRTL(i)$
      if $CritMatrix(i) == 1$ then
          if $t > A \times CMapRTL(i)$ then
              This Block is hotspot!
          end if
      else
          if $t > B \times CMapRTL(i)$ then
              This Block is hotspot!
              $CritMatrix(i) = 1$
          end if
      end if
  end for
```

scheme. Starting from an initial mesh granularity (*initL*) in each iteration (*i*) we increase spatial resolution for on-die areas which are suspected to contain hotspots. Indeed, in each iteration, we examine the temperature map for the current thermal floorplan. For every floorplan block with a temperature value higher than TH we break that block and all of its adjacent blocks into $M \times M$ equal sized smaller blocks. The process will continue until the finest spatial granularity ($finL$) is reached. $initL$, M and $finL$ are constants mainly defined by the user. They should be selected according to the chip die area and the desired spatial resolution and the accuracy with which detection of hotspots should be done. Figure 3.3 shows an example output of our multi-granularity

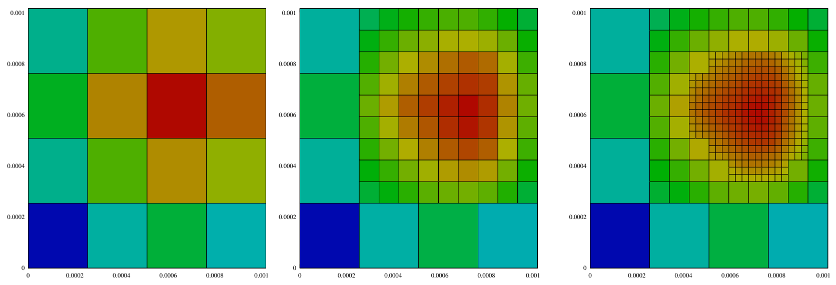


Figure 3.3: Increasing spatial resolution of thermal floorplan for the area of interest. Example, one FFT unit implemented in 40nmLVT running at 800MHz.

thermal simulation for three continuous iterations. Psudocode 4 shows the algorithm in more detail.

MiMAPT has the capability of updating delay and power estimates ($PGate(i, j)$) of the circuit according to obtained non-uniform temperature map ($TGate(i, j)$) during each iteration of simulation. At the first iteration, when gate level analysis is triggered, the temperature map obtained at RTL ($TRTL(j)$) will be used to update the power map of the chip. The updated power map is used for gate level thermal estimation. By the increase in spatial resolution of thermal simulation the accuracy of estimated power at each iteration will increase as well. This feature, also allows MiMAPT to account for

Algorithm 4 Multi-granularity Thermal Simulation

```
Require: initL: initial size of floorplan blocks
Require: finL: highest desirable spatial resolution
Require: TH: threshold value for hotspot temperature
  $minSize \leftarrow initL$
  while $minSize > finL$ do
      (Calculate gate-level power)
      (Do gate-level thermal simulation)
      for each block in floorplan do
          if $T > TH$ then
              (Divide block & adjacents into smaller blocks)
              (Update floorplan)
          end if
      end for
      $minSize \leftarrow$(Minimum block size in floorplan)
  end while
```

the self-heating effect of the chip during its analysis.

3.2.3 Merged Virtual Chip Analysis

MiMAPT allows designers to perform early evaluation when the design is available as set of stand-alone IP blocks but not as the final implemented chip. Figure 3.4 shows how MiMAPT performs early analysis of designs containing several IP blocks. Each IP unit is available as a separate entity, containing its design data, stress patterns, sample synthesized and place&routed chip. Some IP blocks may be available as gate level designs only. Some other IPs may not be available yet, but the designer has an approximate idea regarding the power consumption of the IP block under typical workloads as well as its dimensions. MiMAPT is able to create a virtual chip containing all of the units and run the complete analysis flow on it.

MiMAPT receives a floorplan containing the coordinates for origin of each IP block in the final chip. It then performs all of the required translations in the RTL power reports, cell placement information and gate level power reports to create a unique database containing the required entries for all of the blocks on the virtual chip. The database will be used by the multi-scale RT and gate level thermal analysis engine (described in Section 3.2.2) to obtain temperature estimates on the design.

The obtained temperature map at each iteration will be passed to temperature variation aware power estimation engine in which the power consumption data for each IP unit will be updated according to its estimated temperature. This operation is done with the aid of the translation unit. For the IP blocks which are available only as gate level design, MiMAPT utilizes the cell level power estimates of the block in conjunction with RT level power estimates of other units to create the power trace required by the thermal floorplan. For the black-box IP units which are available only as averaged power density, this value will be used for power calculation during the entire flow.

3.3 Thermal Impact Exploration

In this section we present the effect of temperature variation on key design metrics. We evaluate the effect of on-die temperature variation on the following key design parameters:

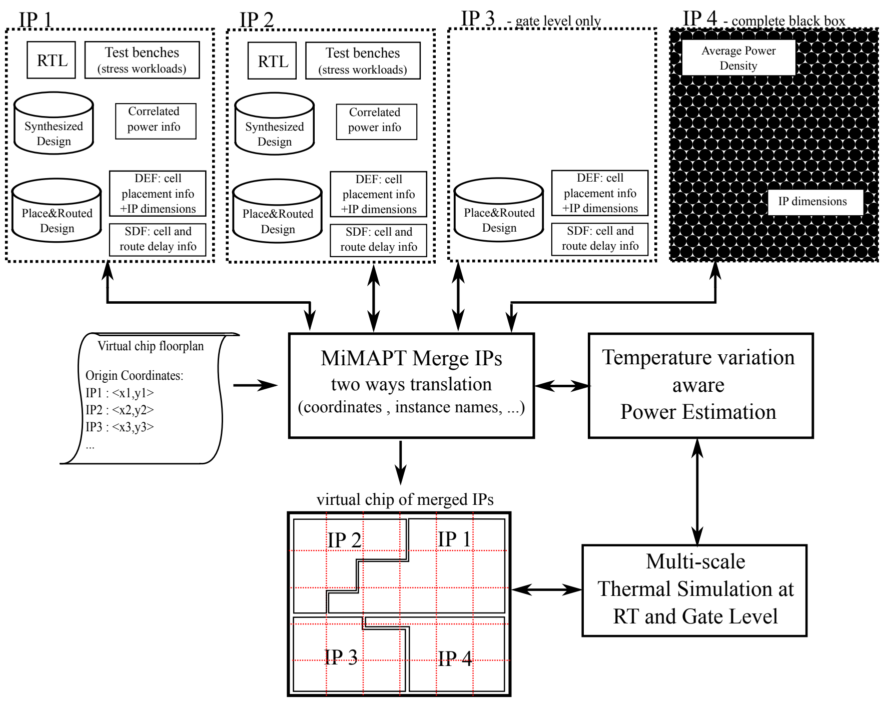

IP 1
RTL
Test benches (stress workloads)
Synthesized Design
Correlated power info
Place&Routed Design
DEF: cell placement info +IP dimensions
SDF: cell and route delay info
IP 2
IP 3 - gate level only
IP 4 - complete black box
Average Power Density
IP dimensions
Virtual chip floorplan
Origin Coordinates:
IP1 : <x1,y1>
IP2 : <x2,y2>
IP3 : <x3,y3>
MiMAPT Merge IPs two ways translation (coordinates , instance names, ...)
Temperature variation aware Power Estimation
virtual chip of merged IPs
Multi-scale Thermal Simulation at RT and Gate Level

Figure 3.4: The operation of MiMAPT during merged virtual chip analysis flow.

- Critical timing paths delay and their physical location.
- Dynamic and static power consumption.

The target design that we use to represent the results needs to meet the following constraints:

- Being representative of a general logic design and being widely used by designers in various hardware platforms.
- Consuming reasonable amount of silicon area and containing an almost balanced number of combinational and sequential elements.
- Being publicly and freely available as source RTL code.

Considering all of the available benchmarks such as [16] and [9], we select Spiral FFT IP block [133] based on the above requirements.

We use two different real-life low-power technology nodes from TSMC® for our tests. Table 3.1 shows key characteristics of these standard-cell libraries. The standard

Table 3.1: TSMC 40nmLP and 65nmLP technology nodes

Std-cell Library	VDD Range (V)	Temp. Range (C)	Cell/Wire Model
TSMC 40nmLP	[0.81, 1.21]	[-40, 125]	CCS/ TLUPLUS
TSMC 65nmLP	[0.90, 1.32]		

temperature range is similar for both technology nodes however, considering supply voltage, 40nm has 8% decrease compared to 65nm. For both of the technology nodes 9 layers of metals is used for power-grid synthesis and design routing.

As described in Section 3.2.1, we use CCS models for leaf-cells delay/power estimations. TLUPLUS [123] models are used to calculate routing delays. Since we use dense power-grids for all sample designs, the amount of IR-drop is negligible. As a result, an equal supply voltage is considered for all design cells during timing/power analysis.

For each technology node, we implement separate chips for each of Low Voltage Threshold (LVT) and High Voltage Threshold (HVT) cells. We compare the behavior of all of the created four chips when imposed to various temperature patterns and under different supply voltage values. Each sample design is passed through the following steps:

1. Multi-Corner Multi-Mode (MCMM) physical synthesis and optimization.
2. Design planning and Power Network Synthesis (PNS).
3. Clock Tree Synthesis (CTS).
4. Placement and Routing (PAR).

We use the latest Synopsys® tools for the current flow.

MCMM physical synthesis is done using four characterized corners. A separate scenario is created for each corner in which timing constraints are defined. Since one of the main goals of our work is to investigate the effect of temperature variation over design speed, we make sure that each design is as optimized as possible in terms of

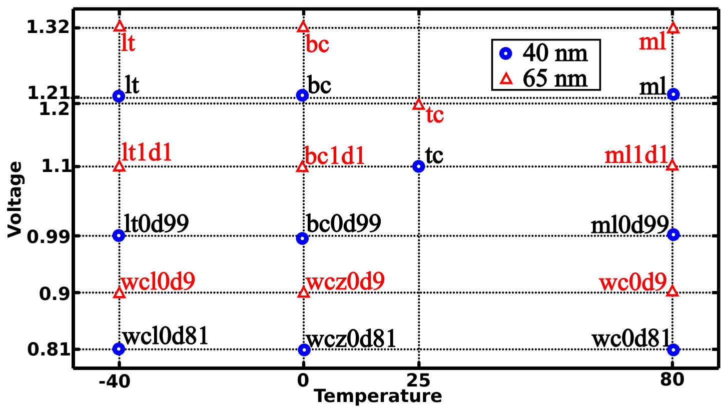


Figure 3.5: Available characterized corners for 40nm and 65nm nodes.

Table 3.2: Area, sequential to combinational cells ratio and number of clock buffers, for each of 4 implemented chips. (Area: mm^2)

Node	VTH	cells area	Die area	seq/comb %	Clk Buf
40nm	LVT	0.4531	1.021	66	734
	HVT	0.4857	1.036	57	11965
65nm	LVT	0.9327	1.985	42	321
	HVT	0.9524	1.969	43	8302

clock speed. Figure 3.5 shows the corner cases which we have available as characterized libraries. `LT`, `ML`, `WC` and `WCL` corners which cover the full temperature and voltage range are used during MCMM flow. We then use the same corners to perform library interpolation, and to obtain design behavior in various operating conditions.

Table 3.2 shows key characteristics of synthesized chips. The averaged leaf-cells area for 40nm chips is approximately $0.46mm^2$ and for 65nm chips around $0.93mm^2$ which is $2X$ larger. Area utilization factor for both of the nodes is approximately 46%. A balanced number (between 40% to 60%) of sequential and combination cells for both of the technology nodes can be seen. As we move from faster LVT to slower HVT cells, number of clock buffers grow to meet timing constraints.

Figure 3.6 shows maximum running clock frequencies of final designs as well as consumed dynamic and static power for the main 4 corners. All values are shown in relative to 65nm HVT chip. To perform timing/power analysis at each corner we use characterized cell library and extracted wire parasitics specific for that corner. For each

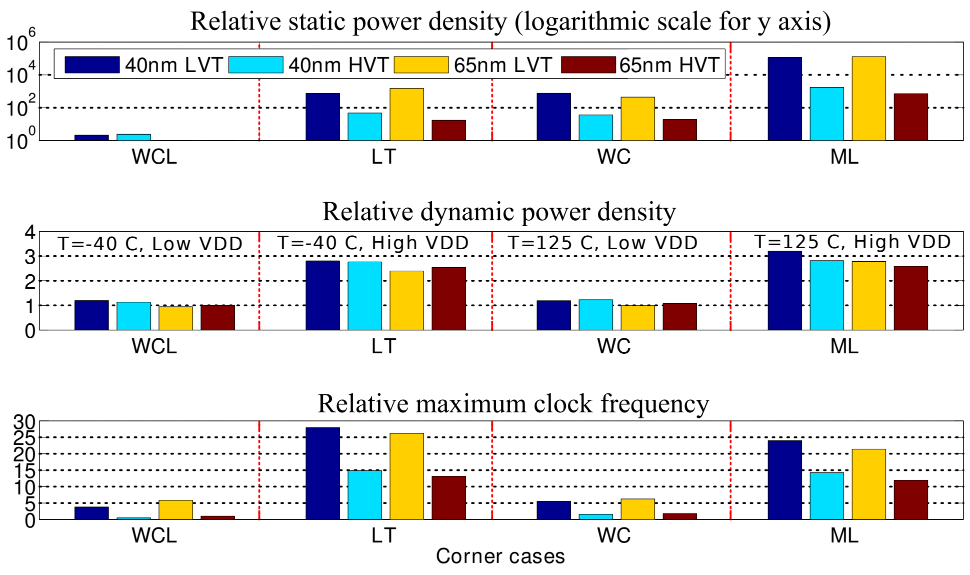


Figure 3.6: Dynamic and static power density values and delay of 4 implemented Chips, for the main operating corners. All numbers are normalized to the values of the 65nm HVT chip at WCL corner for which static power density is 2×10^{-4} W/cm^2, dynamic power density 2.79×10^{-3} W/cm^2/MHz and maximum clock frequency 61 MHz.

design we perform gate level simulation and we apply a unique work-load to the design to obtain switching activities which will be used for power estimation.

As we see in Figure 3.6, the fastest running frequency of the design happens at LT corner which has the lowest temperature and highest supply voltage. At WCL, we see slowest clock for HVT designs. Highest leakage power densities can be seen at ML corner. The fact that for LVT chips the leakage current at ML and LT is slightly higher for 65nm compared to 40nm can be explained by the higher supply voltage of 65nm design. For the rest of the cases, the leakage power density at 40nm is always approximately $2X$ bigger than 65nm. Considering dynamic power density, for 40nm designs it is always larger than the corresponding 65nm design.

We quantify the sensitivity of each design to changes in temperature. Table 3.3 shows the relative sensitivity of estimated power/delay values to changes in temperate. Here for each of the static/dynamic power density values and the clock period, we keep the supply voltage fixed, and we calculate the rate of change in the parameter by the increase in temperature (from -40 °C to 125 °C). In this table, the values are shown as percentages relative to each parameter's final value at 125 °C. For static power we show the absolute values.

Looking at clock period sensitivity, we see negative values for low supply voltage and positive ones for high supply voltage. Indeed due to temperature inversion effect, clock period decreases by the increase in temperature for lower supply voltages thus the design gets faster.

Furthermore, in low supply voltages, the sensitivity of clock period to temperature is orders of magnitude higher than high supply voltage values. Among these, the HVT designs are always the most sensitive to temperature. 40nm designs are always more sensitive than 65nm at low supply voltages. At high supply voltages however, 65nm designs are more sensitive and LVT designs have the highest sensitivity to temperature. Considering static power, we clearly note that HVT designs have always the least sensitivity to changes in temperature. Looking at dynamic power, the relative sensitivity to temperature is always very small and below 0.09%.

Based on represented results, we conclude that the behavior of delay/power in different circuits is heavily dependent on temperature and supply voltage concurrently and it does not necessarily change in a monotonic way. As a result, accurate estimations of delay/power values should be done considering temperature and supply voltage simultaneously.

Table 3.3: Sensitivity of static/dynamic power and delay to changes in temperature for fixed supply voltage. (Change over 1 °C, relative to final value at 125 °C in percents. Static power absolute change value is in $(\mathrm{W/Cm^2})/\mathrm{C}$)

Node	VDD	VTH	Sensitivity to Temperature		
			Static (Absolute)	Dyn. %	Period %
40nm	0.81	LVT	9.10E-04	0.00	-0.28
		HVT	4.17E-05	0.05	-1.30
65nm	0.9	LVT	5.35E-04	0.02	-0.04
		HVT	2.20E-05	0.04	-0.48
40nm	1.21	LVT	1.39E-01	0.07	0.09
		HVT	2.05E-03	0.01	0.03
65nm	1.32	LVT	1.55E-01	0.09	0.11
		HVT	8.44E-04	0.01	0.06

Table 3.4: Estimated delay/power values for BC corner when using characterized library compared to library interpolation

Tech.	VTH	Characterized		Interpolated		Error %	
		Delay	Power	Delay	Power	Delay	Power
40nm	LVT	0.838	0.0221	0.832	0.0209	0.70	5.43
	HVT	1.951	0.0221	1.932	0.0216	1.01	2.26
65nm	LVT	0.844	0.0445	0.839	0.0441	0.62	0.90
	HVT	1.987	0.042	1.973	0.0422	0.74	0.48

We quantify the effect of temperature variation on power and speed of the target chips by creating the following two different test scenarios:

1. Temperature is uniform across entire die and sweeps from 0 to 100 °C. Here we recognize how the design behaves across different temperature values.
2. Non-uniform on-die temperature distribution is a specific user-defined pattern. Here we quantify the error caused by not considering on-die temperature variation on estimated delay/power values.

To accomplish the above objectives, we need first to measure the following items:

1. Accuracy of operating corner interpolation done using CCS models for standard cells.
2. The effect of temperature on the delay of wires.

To quantify the accuracy of operating corner interpolation for standard cells, we use four characterized boundary corners (as shown on Figure 3.5, `ML`, `LT`, `WC` and `WCL`) to perform delay/power estimation for sample chips at *Best Case* (`BC`) corner point. Next, since we have a separate characterized library at BC point itself, we use only this library to estimate delay/power again. During both of the tests, routing delays are kept the same (typical corner).

Table 3.4 shows the results. As we can see in table 3.4 the amount of difference in estimated delay values is always below 2% and for power values lower than 6%.

The distributed nature of routing resources inside the chip, makes temperature variation aware delay analysis for wires difficult. As a result, we use extracted parasitics at typical corner point during the entire wire delay calculations. Thus we need to quantify the effect of this assumption on the accuracy of obtained results.

We extract RC parasitics of the final routed design for all of the available corners. For each corner we then perform delay/power estimation using the characterized standard cell library and extracted parasitics for that specific corner. We then perform delay/power estimation again using characterized standard cell library for that corner, but with extracted parasitics from typical corner and measure the difference in estimated delay and power values. For power, the difference is always negligible. For delay, the difference between estimated clock periods is always below 3%.

We evaluate the behavior of each chip with the sweep in uniform on-die temperature from 0 °C and 100 °C. Table 3.5 shows the results. Dynamic power, changes approximately linearly with temperature. Changes of static power density is exponential thus, we represent its absolute value.

For each (T, VDD) test point, we obtain the physical on-die location of first 10 timing critical paths. In table 3.5, we represent the amount of displacement for the first timing critical path by the change in temperature. For 40nmLVT ($VDD = 1.21$) design we have significant movement of timing critical path.

These changes emphasize the fact that the delay of leaf-cells used in the design do not show similar behavior to changes in temperature. Figure 3.7 shows how the physical locations of first 10 timing critical paths change by temperature. For 40nmLVT a significant jump happens between 75 °C and 100 °C. For 65nmLVT the location of critical paths is different in 75 °C compared to rest of the cases.

Table 3.5: Changes in power and delay with temperature sweep from 0C to 100C. A unique temperature is assumed for entire die.

Node	VDD	VTH	Dynamic (W/cm2/MHz)			Static (W/cm2)		Frequency (MHz)			Disp. %
			0	100	rate (*e-06)	0	100	0	100	rate	
40nm	0.81	LVT	1.00E-01	1.00E-01	0.0000	3.13E-03	8.47E-02	257.73	324.68	0.67	0.40
		HVT	9.81E-02	1.03E-01	48.0406	1.20E-03	5.56E-03	41.03	78.80	0.38	5.89
	1.21	LVT	2.35E-01	2.59E-01	242.7720	8.72E-01	1.43E+01	1639.34	1562.50	-0.77	55.42
		HVT	2.33E-01	2.35E-01	20.5888	3.01E-02	2.41E-01	892.86	884.96	-0.08	0.00
65nm	0.9	LVT	8.01E-02	8.26E-02	25.0170	1.61E-03	4.89E-02	364.96	380.23	0.15	1.99
		HVT	8.61E-02	8.92E-02	31.4994	5.46E-04	2.31E-03	69.49	99.50	0.30	0.00
	1.32	LVT	2.08E-01	2.28E-01	192.9881	1.43E+00	1.69E+01	1538.46	1351.35	-1.87	2.34
		HVT	2.14E-01	2.16E-01	20.9996	1.12E-02	9.70E-02	787.40	740.74	-0.47	0.00

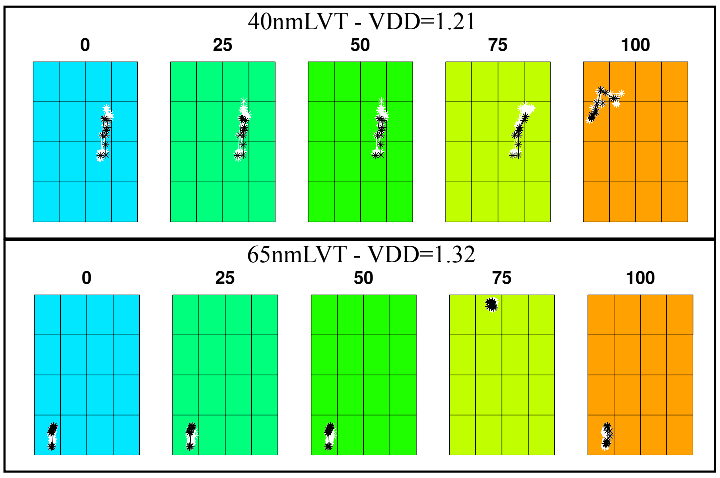


Figure 3.7: Physical location of first 10 timing critical paths of the design for different uniform temperature values. First path is shown in black.

We apply a collection of non-uniform temperature maps to our chips. The following temperature patterns are utilized:

1. *Chess-board* : we divide die area into equally sized rectangles. Two temperature values are selected and applied to them. This case is representative of many-core platforms like Intel SCC [60] in which a large number of CPU cores reside one chip.

2. *Gradient* : we create horizontal gradients of chip temperature across the die. This case is representative of many real-world situations such as PCBs in which components with high power densities (such as DRAM modules) are located just at one side of the CPU or cooling units are placed asymmetrically.

For each of the mentioned patterns we represent the results for two test cases: $G1$ and $G2$.

1. Chess-board: $G1$ pattern uses 40 °C and 60 °C for adjacent blocks. $G2$ uses 20 °C and 80 °C. $G1'$ and $G2'$ patterns use the same temperature values as $G1$ and $G2$ but the location of blocks is reversed.

2. Gradient: $G1$ is a horizontal gradient from 35 °C to 65 °C and $G2$ is from 20 °C to 80 °C. For $G1'$ and $G2'$ the direction is reversed.

For all of the produced temperature maps, the average on-die temperature is always 50 °C. $G2$ and $G2'$ patterns are used as extreme cases of non-uniform on-die temperature distribution.

Table 3.6 shows the results of this experiment. In this table we quantify the amount of error caused by assuming a uniform averaged temperature for entire die and not considering the effect of temperature variation on key design parameters. We calculate the difference (relative error) between each estimated design parameter (considering temperature non-uniformity) and the same parameter with the assumption of uniform temperature of 50 °C. For dynamic power densities, since it changes linearly with temperature the error values are very small (highest error: 0.7%). For static power, significant error values ($> 20\%$) can be seen for $G2$ and $G2'$. This is due to exponential relationship of leakage and temperature.

The average relative error value of $G2$ pattern for all 40nm designs is 15.89% and for 65nm is 13.03%. The higher error value for 40nm design emphasizes the fact that by going into smaller scales of fabrication, the impact of temperature variation on design parameters gets more severe.

Considering clock period, error values of up to 16% can be seen for HVT designs at low supply voltages. For high supply voltage values however, LVT chips show higher percentage of error than the rest. This is inline with our previous observations (Table 3.3). As an example, for 40nmHVT chip at $VDD = 0.81$ V maximum clock frequency of 54.8 MHz is estimated when entire die has a uniform temperature of 50 °C. However considering $G2'$ pattern, the real maximum clock frequency is just 45.7 MHz.

Considering the displacement of first timing critical path, it happens always for 40nm design at low supply voltage. For high supply voltage however, it happens mostly for LVT designs.

Figure 3.8 shows how the on-die physical location of timing critical path changes due to different temperature patterns and different supply voltages. The figure is showing two designs: 40nmLVT and 65nmLVT. As we see, for low supply voltage, the critical path usually happens in colder areas of the chip however, with the same temperature pattern, at high supply voltage, it happens in hotter areas.

Table 3.6: Relative error values for estimated power, minimum clock period and physical location of first timing critical path, when considering non-uniform on-die temperature compared to assumption of uniform averaged temperature for entire die.

Pattern	Node	VDD	VTH	Static Power Error %				Clock Period Error %				First Timing Critical Path Disp. %			
				G1	G2	G1’	G2’	G1	G2	G1’	G2’	G1	G2	G1’	G2’
Chess	40nm	0.81	LVT	3.61	27.23	4.39	28.66	1.69	6.09	1.69	5.86	76.6	76.6	48.59	48
			HVT	0.64	6.03	0.57	5.78	5.85	16.44	7.67	15.70	9.04	9.04	14.15	16.21
		1.21	LVT	2.46	20.16	3.41	21.43	0.16	0.27	0.15	0.94	0	1.39	0	54.63
			HVT	1.37	12.04	1.35	11.99	0.01	0.22	0.08	0.08	0	0	0	0
	65nm	0.9	LVT	0.09	26.33	0.09	26.69	0.45	1.33	0.41	0.40	0	0	73.46	71.55
			HVT	0.30	5.84	0.00	6.69	2.78	9.09	3.50	9.89	78.04	78.18	0.07	0
		1.32	LVT	2.97	11.78	2.97	12.06	1.02	3.36	1.12	3.41	67.29	67.29	0	2.34
			HVT	0.44	11.87	0.74	12.62	0.58	0.46	0.57	1.73	0	23.45	0	0
Gradient	40nm	0.81	LVT	6.79	20.07	2.23	12.42	1.13	2.25	2.91	6.12	1.29	0.4	47.41	47.21
			HVT	2.43	6.47	0.94	0.07	4.84	0.08	9.80	16.54	14.05	70.35	14.25	0
		1.21	LVT	4.81	14.66	1.49	8.76	0.22	0.69	0.38	1.23	0	6.14	52.55	53.94
			HVT	3.03	9.28	0.27	4.15	0.00	0.03	0.03	0.12	0	0	0	0
	65nm	0.9	LVT	0.28	13.61	1.20	14.92	0.67	1.33	0.20	0.40	0	2.77	72.75	72.04
			HVT	0.37	2.12	0.30	3.39	5.17	9.89	4.10	8.32	0.14	0	73.19	73.26
		1.32	LVT	3.17	3.65	2.03	5.46	1.60	3.36	1.69	3.41	67.29	67	0	0.71
			HVT	0.41	5.38	1.23	6.87	0.30	0.42	0.29	0.57	0	61.36	0	0

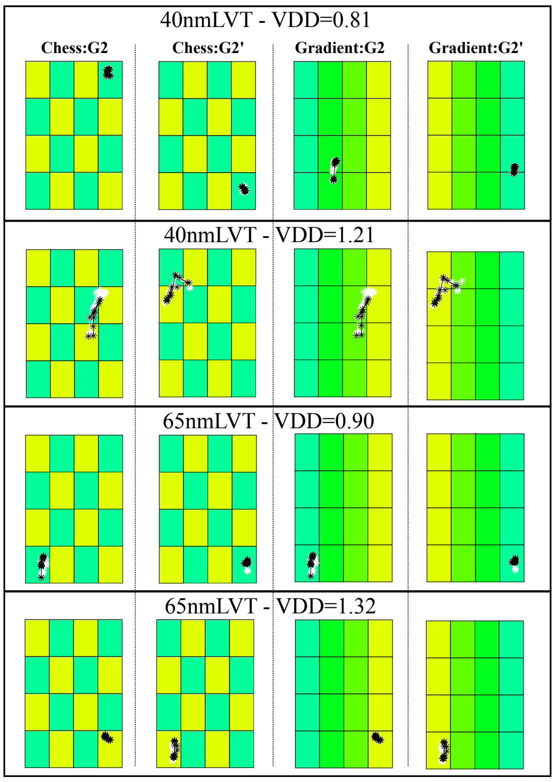


Figure 3.8: Location of first 10 timing critical paths for different temperature maps and supply voltages. First critical path is in black.

3.4 Adaptive Hotspot Detection at RTL

In this section, we evaluate our adaptive hotspot detection algorithm at RTL. Based on real power values, we create an ensemble of virtual gate level and RTL power maps by adding Gaussian random variables to per-floorplan block power values at gate level. We change random variable's characteristics to simulate different situations of RTL power estimation.

For each power map at gate level we create 10 different power maps at RTL by changing the mean of the Gaussian random variable ($\mu = \{-0.2, -0.1, 0.0, 0.1, 0.2\}$). We compare the output of thermal simulation for all of the gate level and RTL power pairs. For each pair we obtain the number of hotspots and their locations at gate level and we compare them with the output of hotspot detection methods at RTL.

Figure 3.9 shows the performance of our adaptive method (*A-Temp*) compared to using a unique user defined threshold value (*TH Only*). In this figure, *(a)* shows the

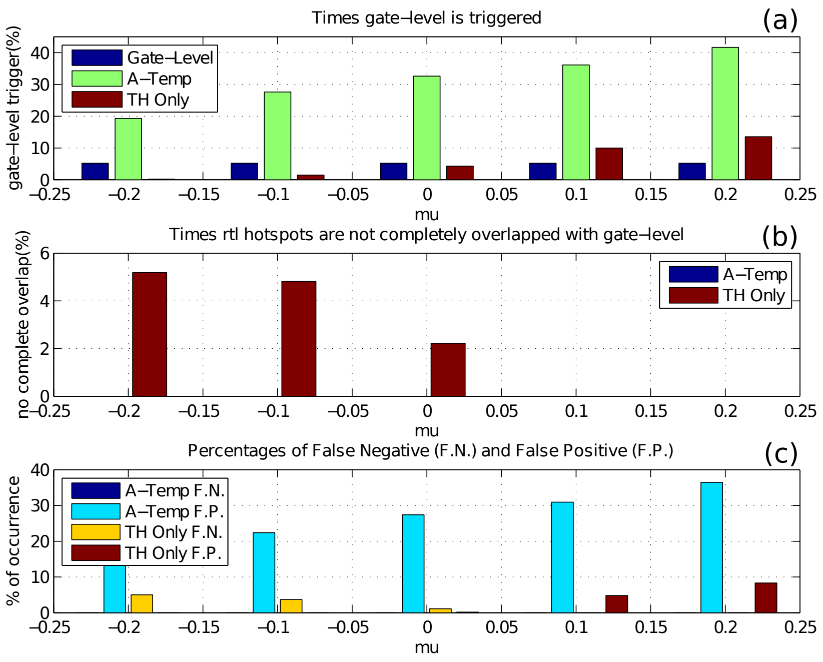


Figure 3.9: hotspot detection methods comparison (Adaptive vs. TH only)

percentage of situations that hotspots exist in the chip and gate level simulation should be triggered, compared to percentage of situations that each of *A-Temp* and *TH Only* trigger gate level simulation. *(b)* shows percentage of cases in which the estimated spatial location of hotspot by each of *A-Temp* and *TH Only* is different than its estimated location at gate level. As we can see, *A-Temp* estimates the spatial location of hotspots correctly in all of the situations. Finally, *(c)* shows the percentage of detected false positives and false negatives for each of *A-Temp* and *TH Only* methods considering all of the 180 test cases. Different than *TH Only*, false-negative for our method is equal to zero which means it captures all of the hotspots completely.

3.5 Example MiMAPT Operation

To evaluate MiMAPT in a real test case, we use the *merged virtual chip creation* capability of MiMAPT, and build a chip containing a grid of 16 FFT 40nmLVT units. Total chip area is $4 \times 4\ mm^2$. This approach (inspired by [113]), allows us to have various power distribution patterns as well as significant on-die temperature variations during

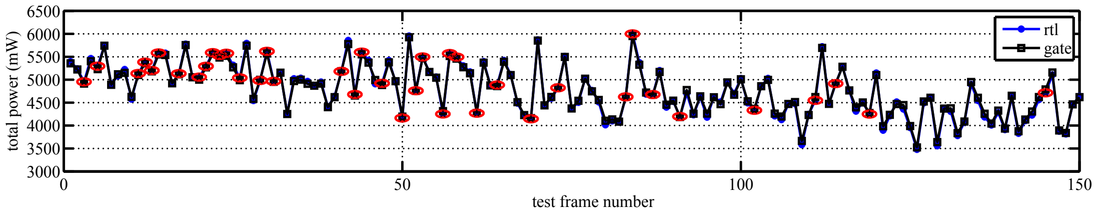


Figure 3.10: Total estimated power for each of the *TF*s at RT and Gate level. The chip contains 16 instances of FFT unit. Red circles show *TF*s for which a suspicious block is detected.

our tests.

We then create 150 different *test frames* (*TF*s) which impose different stress workloads and running clock frequencies (400 to 625 MHz) on units of the chip.

Duration of each *TF* is 0.2 seconds. Figure 3.10 shows the estimated total power for each of the *TF*s at RT and gate level. Typical operating condition is used for power estimation. The total power varies between 3.5 W to 6 W among different *TF*s. As we see, the estimated power at RTL tracks the gate level power with a very good accuracy. The maximum difference between gate level and RTL power is 2.2%.

We execute MiMAPT for all *TF*s and store spatial and temporal information related to detected hotspots. Then, we perform power and temperature estimation of the design without MiMAPT for the same set of *TF*s at gate level and at the finest level of granularity. We perform comparison between MiMAPT and fine-grain analysis results for a threshold value of $TH = 90.0$ °C. During this experiment, the target is to study the speedup of MiMAPT over fine-grain method thus the temperature variation aware (TVA) power/delay re-calculation feature of MiMAPT is off. RTL floorplan and the initial floorplan at gate level are 12×12 blocks (*initL*). *A* and *B* coefficients in RTL hotspot detection are 0.99 and 1.3 respectively. For increasing spatial resolution of the thermal floorplan at gate level, we divide each hotspot block into 2×2 equal sized smaller blocks ($M = 2$).

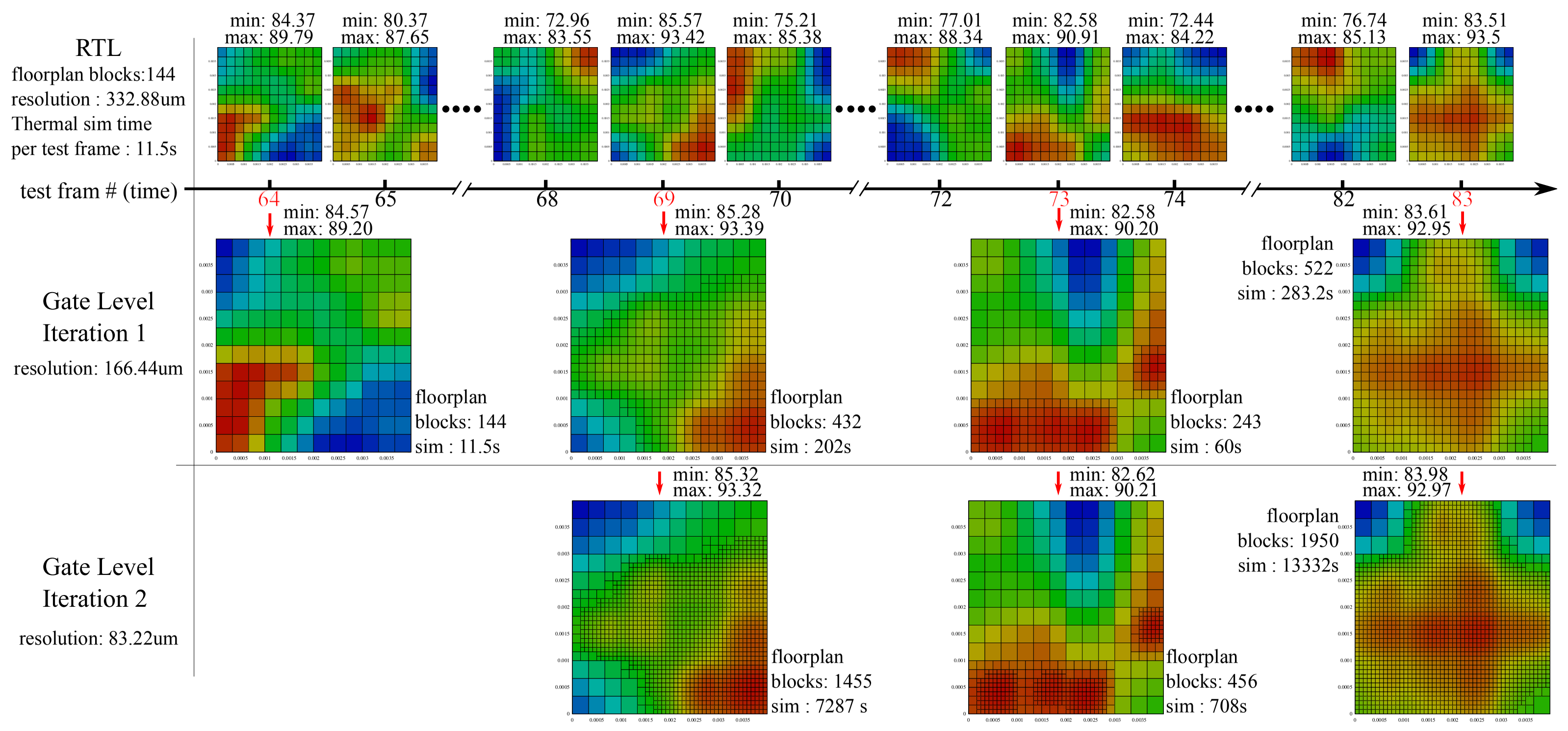


Figure 3.11: Sample temperature gradients obtained during MiMAPT analysis on the sample chip.

We represent the results in terms of execution time and accuracy of detecting hotspot location and temperature. Among 150 available *TF*s, for 39 of them MiMAPT detects critical blocks at RTL and triggers gate level thermal analysis. For the other 111 *TF*s, gate level is not triggered, thereby saving considerable run-time. Among critical *TF*s at RTL, 15 of them are false positives. Meaning that when accurate gate level analysis is executed for them no thermal hotspot is detected.

When gate level is triggered MiMAPT performs three iterations of thermal simulation to achieve required spatial resolution of 83.22 um. Figure 3.11 shows the operation of MiMAPT for several *TF*s. The first row shows obtained temperature gradients during RTL thermal simulation. The maximum and minimum temperatures for each of the shown gradients is written at the top of it. Critical *TF* numbers are highlighted in red. The second and third rows show the obtained temperature gradients during second and third iterations of gate level thermal simulation. As we see, *TF* number 64 is a false positive. *TF* numbers 69, 73 and 83 contain real hotspots and the resolution of thermal floorplan increases for them during each iteration. The finest level of granularity ($finL$) is 83.22 um.

We then perform thermal simulation using only the fine-grain floorplan which contains total 2304 blocks. Comparing the estimated temperature for hotspots using MiMAPT and the fine-grain method, The difference is around 0.02 °C. For every hotspot block at fine-grain, there exists a corresponding block in MiMAPT which has the same location and size and is announced as hotspot. As a result, the distance between location of hotspots detected by MiMAPT and fine-grain is zero. MiMAPT detects all of hotspots with a very good level of accuracy, thus there is no false negatives.

Table 3.7 contains the averaged execution times for each step of thermal analysis flow. Equation 3.1 shows the total execution time of MiMAPT.

$$\begin{aligned}
t_{RTL} &= \sum_{TF=1}^{N} (t_{sim}^{RTL}(TF) + t_{pwr}^{RTL}(TF) + t_{thr}^{RTL}(TF)) \\
t_{Gate} &= \sum_{TF\in N'} (t_{sim}^{Gate}(TF) + t_{pwr}^{Gate}(TF) + \sum_{i=1}^{K} t_{thr}^{Gate}(TF,i)) \\
t_{tot}^{MiMAPT} &= t_{RTL} + t_{Gate}
\end{aligned} \tag{3.1}$$

In these equations N is total number of *TF*s and N' is the list of *TF*s detected as critical

Table 3.7: Definition of symbols representing execution times of different tasks done during analysis of one *TF*, and their average value for the example chip.

Symbol	Description	Average Value
t_{sim}^{RTL}	RTL Logic simulation	2.46
t_{sim}^{Gate}	Gate level logic simulation with SDF	686.5
t_{pwr}^{RTL}	Power estimation at RT level	14.8
t_{pwr}^{Gate}	Power estimation at Gate level	62.13
t_{thr}^{RTL}	Thermal simulation at RT level	11.5
$t_{thr}^{Gate}(1)$	Thermal sim. at Gate level, first iteration	11.5
$t_{thr}^{Gate}(i)$	Thermal sim. at Gate level, *i*th iteration	variable
$t_{thr,Fine}^{Gate}$	Thermal sim. at Gate level, fine grain	18439

at RTL. K is the number of iterations for which we do thermal simulation for one *TF* at gate level (false positives: $K = 1$, others: $K = 3$).

$$t_{tot}^{Fine-grain} = \sum_{TF=1}^{N} (t_{sim}^{Gate}(TF) + t_{pwr}^{Gate}(TF) + t_{thr,Fine}^{Gate}(TF)) \tag{3.2}$$

Total execution time of fine-grain method is shown in Equation 3.2.

Table 3.8 shows the execution time of MiMAPT compared to fine-grain. Considering all the 150 *TF*s, total execution time at fine-grain is 2878×10^3 seconds. In contrast, total execution time for MiMAPT is 92×10^3 seconds. In total, MiMAPT is 31X faster than fine-grain at the same level of accuracy. It should be noted that, even if we disable the multi-granularity floorplan definition of MiMAPT, and for each critical block at RTL initiate the gate level analysis at finest resolution directly, MiMAPT is still 3.67X faster than the fine-grain method, thanks to its early thermal analysis at RTL.

In order to provide a better perspective on the range of possible speedups for MiMAPT, we calculate average MiMAPT time when all *TF*s are either non-critical, false positive or critical frames. For our sample chip, for an assumed experiment in which every *TF* is non-critical, MiMAPT reaches the maximum speedup of 667X. However when every *TF* is false-positive, the speedup decreases to 24X. Finally if every *TF* contains hotspots, considering an averaged execution time for each of the

Table 3.8: Execution times comparison: MiMAPT vs fine-grain. (Time in seconds$\times 10^3$)

	RTL			Gate		
Method	logic sim.	power est.	thermal sim.	logic sim.	power est.	thermal sim.
MiMAPT	0.369	2.22	1.72	26.77	2.42	58.55
Fine-grain	-	-	-	102.9	9.31	2765.8

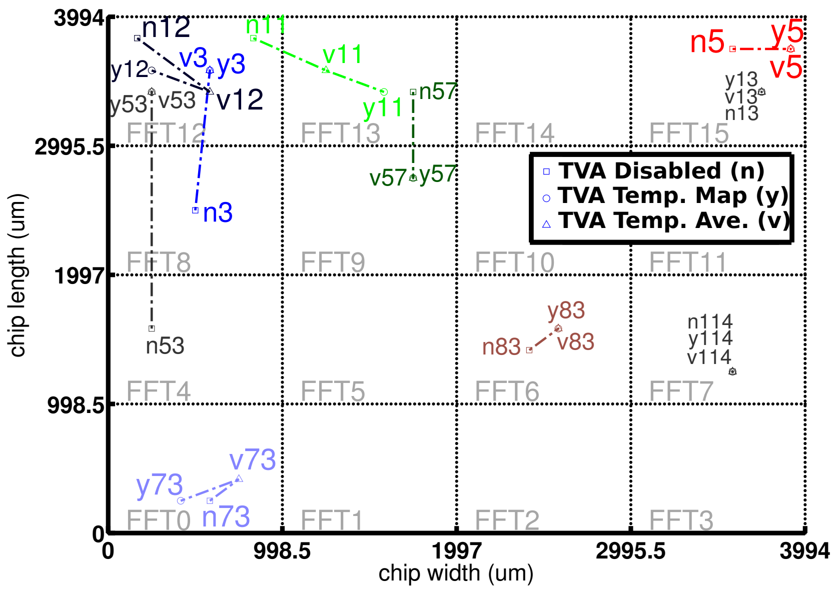


Figure 3.12: Physical coordinates of hottest point of the chip when TVA is disabled (shown with n letter), TVA is active and temperature map is used (y), TVA is active and averaged die temperature is used (v).

iterations of gate level thermal simulation, the speedup degrades to a lower bound of $8.3X$.

We now enable the temperature variation aware (TVA) power/delay analysis feature of MiMAPT. As described in Section 3.1, this makes MiMAPT consider the effect of self-heating and on-die temperature non-uniformities during its analysis. As a result, MiMAPT updates the estimated power of the design at the beginning of each iteration according to the obtained temperature map during the previous iteration. It also updates the reported timing/delay of the circuits. To quantify the impact of this method, we select 10 *TFs* which contain hotspots and for each of them we compare the location and temperature of the hottest design points and also the highest reported design frequency for these three cases:

1. Normal method in which the TVA analysis is off.
2. TVA is on however, the averaged chip temperature is used to update the power map.
3. TVA is on and the full temperature map obtained from the previous step is used to update the power map.

Figure 3.12 shows example physical locations of the hottest point of the design for each of the *TF*s. Each set of points are indicated with their *TF* number. The points marked by the *'n'* letter show the results when TVA is not active. The *'v'* points are when averaged die temperature is used and *'y'* letters show when full temperature map is utilized. As an example, for $TF = 12$, we notice 588.4 um difference in the locations of detected *'n'* and *'v'* points. Moreover, a distance of 372.1 um can be seen between corresponding *'v'* and *'y'* points. As shown, the physical location of the detected hotspot point can change significantly when TVA analysis is taken into consideration. Even usage of averaged temperature value for entire die can lead to inaccurate results. Comparing the estimated temperature values for *'v'* and *'y'* points, differences of more than 0.8 °C can be seen. Looking at the maximum running frequency of the design for each of *'v'* and *'y'* cases, we notice differences of more than 20 MHz (3.5% of maximum design running frequency). Considering the leakage power, differences of up to 30% can be observed for estimated leakage power of design units in each of *'v'* and *'y'* cases. The presented results highlight the importance of taking temperature variation into account while doing power, delay and thermal analysis.

Chapter 4

Temperature Variation Aware Energy Optimization in 3D MPSoCs

In this Chapter, we perform an in-depth study on performance, timing, power and leakage of 3D stacked Wide-I/O DRAMs and track their key parameters with temperature change [3, 8].

Through careful evaluation of temperature distributions in a 3D IC with Wide-I/O DRAMs, we propose architectural enhancements for the DRAM subsystem which improve energy consumption.

In order to assess and quantify the advantages of our proposed ideas, we build a suitable virtual infrastructure which considers all key characteristics of a 3D MPSoC with Wide-I/O DRAMs in detail.

First we study the academic research related to DRAM memories, stacked DRAM memories in 3D ICs and energy optimization in them. Then we illustrate the developed TLM environment (Section 4.2), and the DRAM (Section 4.2.1), CPU (Section 4.2.2) and thermal (Section 4.2.3) models. Then we discuss the temperature variation aware management of refresh rates for the 3D Wide-I/O DRAM (Section 4.2.4). The design and operation of a thermal controller is disscussed in Section 4.2.5. The adaptive sampling method to improve simulation speed is described in Section 4.2.6. The experimental results are presented in Section 4.3. Our feasibility study on hardware acceleration of the thermal simulation is discussed in Section 4.4.

4.1 Related Research Works

Two integrated performance, power and thermal modeling infrastructures for evaluation of energy and thermal management policies are presented in [115] and [97]. Both of the tool-sets mainly integrate a group of simulation software (*gem5*, *DRAMSim2*, *Hotspot* and etc.) to perform performance, power and thermal modeling. The advantages of our infrastructure over these tool-sets are the followings:

1. We use TLM models to integrate different functional units of MPSoC together. The TLM environment provides us a high level of flexibility in creating various hardware structures and performing simulations.

2. Our infrastructure is tailored to the simulation of 3D-integrated Wide-I/O DRAM memory systems and tracks the timing and power of Wide-I/O DRAMs with the changes in temperature. This is contrary to other papers which rely on common DRAM models (e.g. DDR3) to estimate the behavior of a 3D stacked DRAM component.

3. Prior publications use adapted versions of 2D thermal simulators to perform 3D thermal estimations. Unlike, we use the 3D-ICE [119] thermal simulator which is inherently designed for 3D chips.

4. To mimic the workloads executed by today's mobile phones, we run Android OS and a set of real-world benchmarks on our multi-core ARM MPSoC. This is opposed to papers, which run benchmarks in the bare-metal or at most Linux environment.

5. We adaptively tune the sampling interval (t_{sim}) of the simulation based on the thermal profile of the chip. We also study the feasibility of hardware acceleration of thermal simulation. These are not addressed in any prior academic research contribution.

DoceaPower introduces Aceplorer[41], an ESL platform for exploring low power/temperature architectures. It integrates with Synopsys Platform Architect environment and allows developers to perform estimations on power and temperature at ESL. In order to have accurate estimations on performance and power, the users must utilize cycle accurate

(CA) or approximately timed (AT) TLM models for CPU cores, memory units and other key components of the system. However, these components are usually available as separate units sold with separate license. Instead our solution practically integrates the freely available *gem5* models into the TLM environment, and prepares highly accurate TLM2.0 DRAM models, providing the same functionality without the need to any separate license.

A 3D MPSoC with Wide-I/O DRAM is presented in [42]. The 3D-IC features thermal sensors which can be used for online monitoring of temperature and tuning thermal models as well. The floorplan developed in this paper for the thermal model is indeed inspired by this work.

A detailed thermal characterization of 3D ICs with Wide-I/O DRAM is presented in [142]. The effect of different alignments for DRAM dies and TSVs in overall chip temperature is evaluated. The paper focuses on thermal modeling only and does not discuss architectural or thermal management ideas.

An exploration of 3D DRAM architecture which results in a highly efficient Wide-I/O DRAM subsystem is presented in [14]. The work however, focuses on architectural issues only and does not consider the effect of temperature variation on the performance metrics of the Wide-I/O DRAM. Indeed, we utilize the DRAM model developed in [14] in our work.

A thermal model and a thermal management policy for DRAM modules is proposed in [90]. As shown in the paper, the thermal control of the DRAM is mainly obtained through decreasing the number of CPU accesses to the memory. The paper discusses a 2D structure however, as we will see, similar to this paper, our main mechanism to govern the temperature of DRAM channels is scaling down the activity level of the CPU cores.

The timing accurate DDR2/3 memory system simulator *DRAMSim2* is introduced in [116]. This tool targets DDR memories and not 3D Wide-I/O DRAMs. The model is cycle accurate (CA), which burdens the integration into TLM environments. In contrast, we can insert our DRAM TLM2.0 model into any TLM environment in which a vast ensemble of other TLM models [132] is available as well.

A well-known and often used power model for DRAM is provided by Micron [69]. This model has certain limitations. First, Micron uses the minimal timing constraints from the data sheet specifications instead of the actual timings. However, there are dependencies between consecutive memory accesses so that the controller may accelerate or postpone commands. Second, Micron assumes that the controller uses a close-page policy and that there is only one bank open at the same time. An improved version of this power model was presented in [30], which uses actual timings from transactions. We use an enhanced version of this power model in our design [11]. The refresh rate of a DRAM device depends on its leakiest cells. However, the number of low retention time cells is relatively small compared to the total number of cells in a DRAM. A detailed study on data retention time in modern DRAM devices was done in [92]. Software approaches to reduce refresh power by retention-aware placement of data in the DRAM are presented in [136]. A method for reducing the total refresh rate by grouping the DRAM rows into different retention time bins and applying different refresh rates on them is presented in [91].

We extend these ideas by studying the relationship between refresh rate of a DRAM bank and its temperature. We show that due to the lateral and vertical temperature variations in 3D MPSoCs, it is not necessary to refresh all banks of a DRAM channel at a similar rate.

4.2 The ESL Virtual Infrastructure

The developed infrastructure which is shown in Figure 4.1 contains five major parts:

1. The *gem5* Environment: it models the operation of ARM CPU cores. The ARM CPU cores simulated by *gem5* are running Android OS and execute a set of well-known real-world benchmarks. *gem5* is configured to capture complete DRAM access traces (after L2 cache). Further descriptions come in Section 4.2.2.

2. The TLM Environment: it models the entire MPSoC. It contains the TLM models for the DRAM memory units and their frontend and backend logic (Section 4.2.1) as well as *gem5* trace players which resemble the operation of CPU cores in the TLM environment by re-playing the traces recorded in *gem5*.

3. Power models: receive the performance statistics of CPU cores and DRAMs from the TLM environment and estimate their power and fill-up the power traces used by the thermal model. Sections 4.2.2 and 4.2.1 discuss these in more detail.

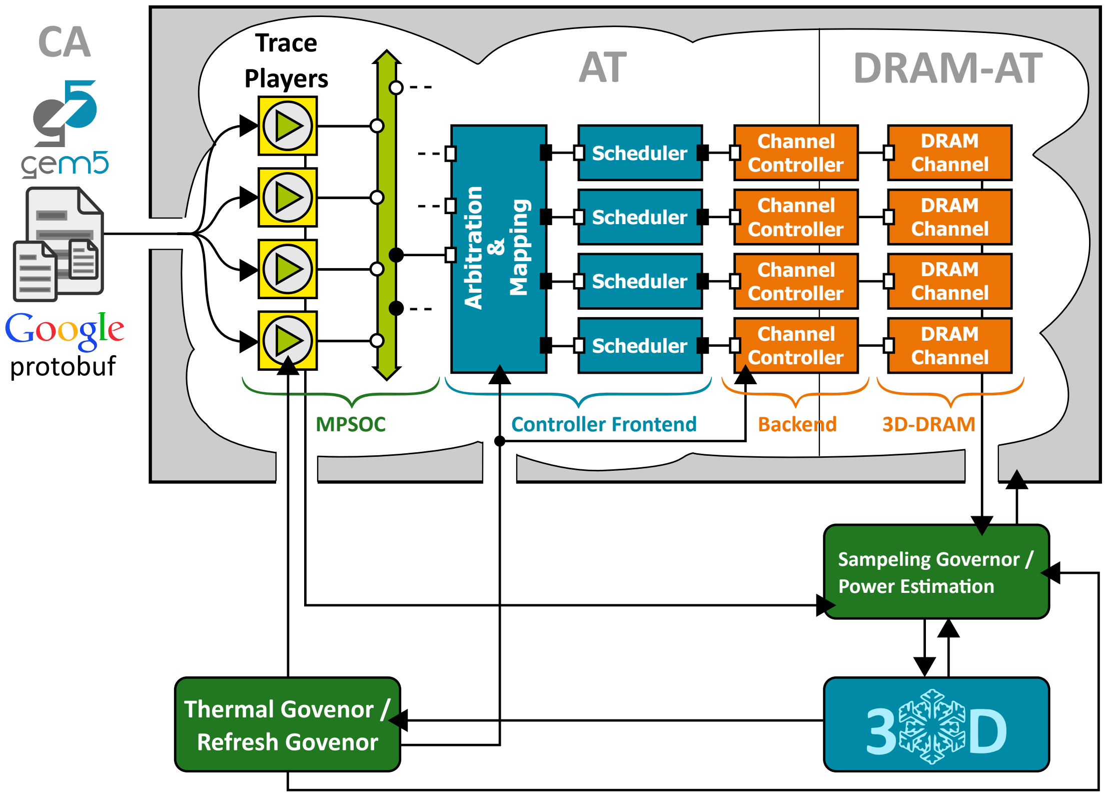


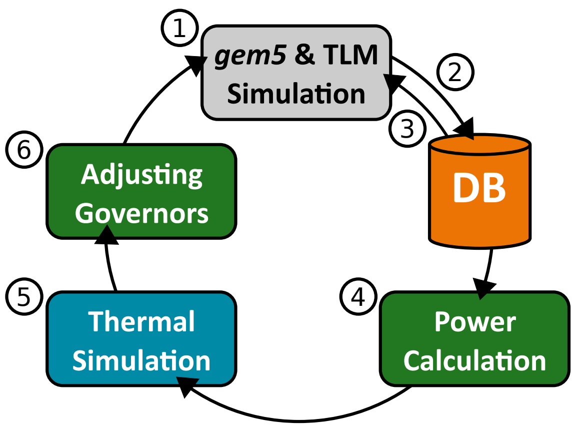
Figure 4.1: Virtual Platform with Thermal Control Loop

4. Thermal model: receives power traces, ambient temperature (T_{amb}), sampling interval and previous thermal profile of the chip, and calculates the thermal profile of the chip (Section 4.2.3).

5. Governors: are three different units to govern the sampling interval of simulation; t_{sim} (Section 4.2.6), the refresh rate of DRAM units (Section 4.2.4) and the chip temperature (Section 4.2.5).

Considering the above descriptions, Figure 4.2 shows the detailed procedure for one simulation step.

The TLM environment is created using the *Synopsys Platform Architect* and system simulations are done using the *Virtual Prototype Analyzer*. This gives us complete control over the execution flow of simulation as well as full access to configuration and statistics registers of our simulated hardware (e.g. trace players and DRAMs).



1. Run for t_{sim}
2. Store #instructions per CPU and #DRAM cmds. per bank
3. Reset counters
4. Calculate per CPU and per DRAM bank power
5. Obtain temp. distribution
6. Set per bank refresh rate, set t_{sim}, set f, V_{DD}

Figure 4.2: Flow of one Co-Simulation step

A set of *TCL* scripts control the flow of simulation and provide the required communication between the TLM environment and the outside world.

A set of python scripts operate in parallel with the TLM environement. At the end of each sampling interval (t_{sim}), the performance metrics of CPU and DRAM are read by *TCL* scripts. A *Python* script is then responsible for estimation of CPU and DRAM power and invokation of the thermal simulation.

The governor units, implemented in Python, receive the simulated temperatures and make decisions on sampling interval of the simulation, refresh rates of DRAM units and the clock frequency and supply voltage of CPU blocks and DRAM channels. In the following, we describe the important parts of the infrastructure in detail.

4.2.1 DRAM Model

The 3D-DRAM memory subsystem used in our platform, consists of a controller frontend, a channel controller and a Wide I/O DRAM model. It is shown in Figure 4.1. The standard TLM AT protocol is extended with a DRAM specific protocol (DRAM-AT) [10] to provide very fast simulation speeds with high timing accuracy. We improve the DRAM power model of [30] to create our DRAM power model for the TLM platform [11].

For this work we extended the model of the DRAM controller to support in addition to the auto-refresh command `REFA`, separate refreshes per bank or groups of banks: `REFB`.

The DRAM power model accepts per-bank activity statistics and generates per-bank power values at each simulation step. This functionality is later exploited by the refresh governor of the MPSoC to assign different refresh rates to the different banks of a DRAM channel according to their temperature.

Table 4.1 represents the relationship between temperature and refresh rates of a DRAM bank (50 nm technology). The refresh rates in this table are calculated based on [102].

Table 4.1 is built taking into account that commercial DRAM products exploit temperature sensors with high level of accuracy (± 1 °C) around the calibrated junction temperature range (90 °C).

The inputs required for calculation of power consumption of one DRAM bank in a channel (c) during one sampling period of simulation are the following: T_{ave} is the temperature of the bank which is used to calculate the static power. f is the running frequency of the DRAM channel and is similar for all of the banks within a channel and $\mathbf{V_{stat}}$ is the input vector containing the activity statistics of the DRAM bank during this sampling interval. It basically contains the number of issued DRAM *ACT*, *PRE*, *RD*, *WR* and *REFB* commands. $\mathbf{V'_{stat}}$ contains global DRAM channel statistics. These include total number of bank activate and precharge commands issued in the background. For a detailed description of the DRAM power model and its validation methodology and results, please refer to [30] [14].

The quantized clock frequency values supported by our DRAM model are the followings: $\{100, 133, 166, 200\}$ MHz. The supply voltage of the DRAM is always fixed at 1.2 V.

Our MPSoC contains 4 channels of Wide-I/O DRAM. Each channel consists of 8 banks and spans 4 DRAM dies. Thus each DRAM die, contains 2 banks per DRAM channel, see Figure 4.3 (Banks 0 and 1 belong to Mem Die 1, banks 2 and 3 belong to Mem Die 2, etc.). Figure 4.4 (a) shows the placement of DRAM banks in each DRAM die. The density of each DRAM die is 2 Gbit. Our virtual MPSoC contains total memory of 1 GBytes. The physical dimensions of DRAM banks and the silicon die are calculated based on 50 nm technology [81]. Section 4.2.3 describes the physical properties of the chip in more detail.

Table 4.1: Refresh rates of DRAM banks vs. temperature

Temp (°C)	35	45	65	85	87.5	88.75	90
Refresh Rate (ms)	192	128	96	64	56	52	48
Temp (°C)	91.25	92.5	95	100	105	bigger than 105	
Ref Rate (ms)	44	40	32	24	16	8	

4.2.2 CPU Model

Our CPU model consists of two distinct parts: the *gem5* simulator and TLM *gem5* trace players. Overall, we first run our benchmarks on the *gem5* simulator and record detailed traces of DRAM accesses (after L2 cache). Inside the TLM environment, we build a complete virtual MPSoC which contains the *gem5* trace players to represent the CPU cores and other necessary glue logic. We then re-play the recorded traces inside the TLM environment to perform different evaluations on temperature, power and energy consumption of the MPSoC. The timings of the traces will be dynamically adjusted at the time of play-back inside the TLM environment according to the timings and latencies introduced by the DRAM models.

gem5 is configured for the ARM ISA and it is run in detailed, out-of-order mode. We run Android 2.3 Gingerbread on this platform and use three real-world well-known benchmarks to stress the CPUs and the memory. Our selected benchmarks are: *AndEBench* [43], *0xBench* [51] and *SmartBench* [12]. In total the benchmarks perform 6.53×10^9 read and write transactions to the DRAM memory.

To configure the hardware architecture simulated by *gem5* we refer to the ARM sub-system of a real-world heterogeneous chip [140] (which contains a dual-core ARM Cortex-A9 MPCore unit) as our reference hardware and use exactly similar architectural parameters. This approach allows us to use the reported power consumption values of the real hardware to create and accurate CPU power model.

In *gem5* configuration, number of cores is set to 2 ($n = 2$), the system contains separate private L1 caches for each of the cores and a shared L2. L1 instruction and data caches are 32 KBytes. Each cache line is 32 *Bytes* and the L1 associativity is set to 4. The size of shared L2 is 512 KBytes, it is 16 way set-associative and each cache line is 32 Bytes. The size of DRAM is equal to 256 MBytes.

We use Google protocol buffers to record memory traces. Each entry in the trace contains: *gem5* tick count in which the memory access has occured, type of access and size of data transfered, access address and total number of instructions executed by each of the two ARM CPU cores up to the this tick point. These information will be used by the AT TLM trace player to generate proper transactions to the memory subsystem. Since *gem5* is simulating a detailed out-of-order model, each trace player may be handling several transactions at the same time.

Our virtual MPSoC contains four *gem5* trace players or equivalently four instances of ARM MPCore units. This is equal to a total number of eight cores. Each MPCore unit has its own running frequency and supply voltage. The total 1 GBytes DRAM memory is evenly divided between 4 MPCore units.

The inputs to the power model of the CPU are: T_{ave}, the averaged temperature of the core to estimate static power, IPC is the average count of instructions executed in one CPU clock cycle, f is the running clock of the MPCore block and V_{DD} is the supply voltage.

To implement the CPU power model we mainly rely on the power model provided by [139] for the ARM sub-system of the device. Using [139] we create look-up tables which contain static power of the cpu core for different temperatures and its dynamic power for different IPC values. Considering the fact that dynamic power is proportional to $VDD^2 \times f$, [50] we update the dynamic power based on the current $VDD(m)$ and $f(m)$.

The relationship between the maximum running clock of an ARM Cortex-A9 MPCore and its supply voltage is described in [82]. According to [82] for the following frequency ranges $f = \{700, 1050, 1300, 1600\}$ MHz the following supply voltages are used: $V_{DD} = \{0.8, 0.9, 1.0, 1.1\}$ V.

To resemble a real-world MPSoC, all of the frequency and voltage values in our system are quantized numbers; clock frequencies are all multiples of 50 MHz and supply voltages are from the V_{DD} array.

4.2.3 Thermal Model

We build our thermal model using the 3D-ICE [119] thermal simulator which supports definition of layers with non-homogeneous materials in the chip stack. Considering the

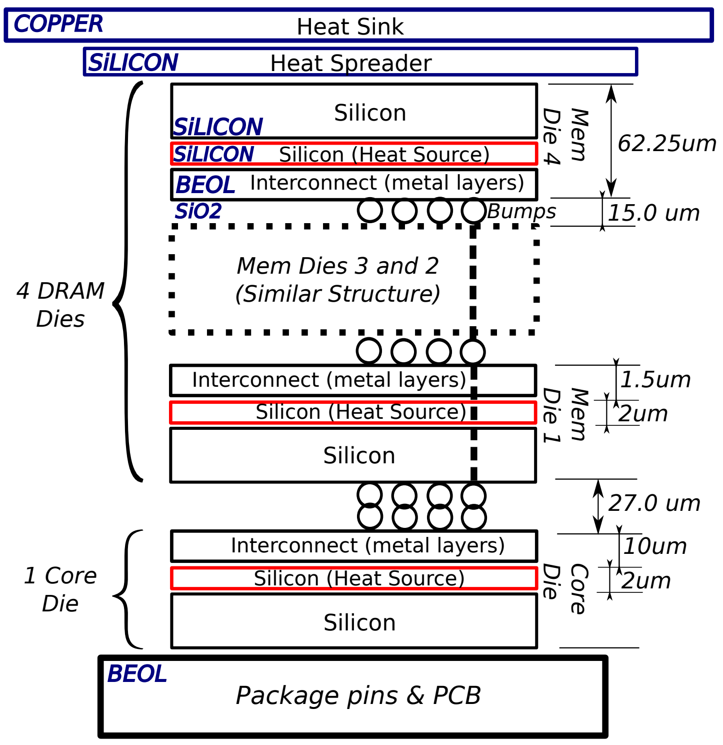


Figure 4.3: 3D stack-up of the virtual MPSoC

fact that, the 3D-ICE engine works in co-simulation with the TLM environment, the initial temperature values for each thermal simulation should be exactly those obtained during the previous thermal simulation. We have further extended 3D-ICE functionality to dump its complete temperature distribution at the end of each simulation and to re-load it in the beginning of the next one.

To create a complete realistic thermal model, real-world numbers are used to define the key dimensions of the 3D structure [81] [129]. Figure 4.3 shows selected values for 3D stackup. The thermal floorplan of each silicon die which is indicative of the size and coordinates of on-die units is shown in Figure 4.4.

As Figure 4.4 shows each silicon die is $8.6 \times 7.4\ mm^2$. For thermal simulation, the chip is divided into equal sized cubes of $0.3\ mm^3$. Each transient thermal simulation is done with a duration of t_{sim} which is decided by the sampling governor unit (described in section 4.2.6). The spatial resolution of the thermal model is fixed during the entire simulation.

According to [81] the TSVs are built using tungsten. We define different *material*

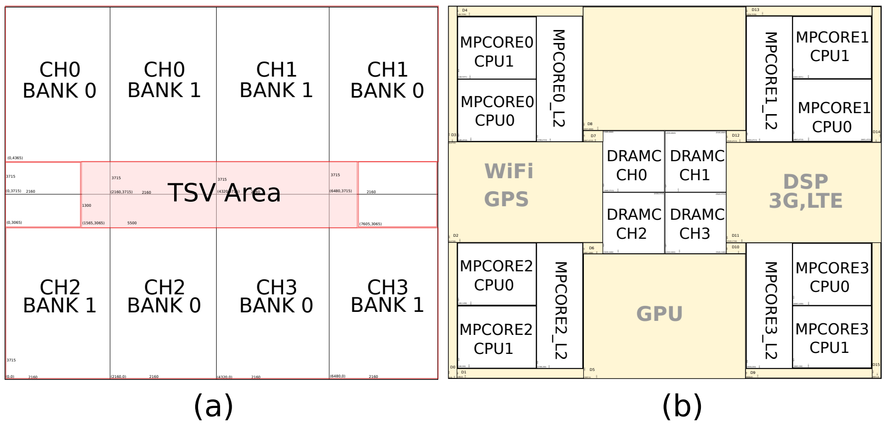


Figure 4.4: Geometries used for thermal model: (a) DRAM die (b) Core die.

floorplan file for each of the layers of our 3D structure. In Figure 4.4 the TSV area of the DRAM die is shown as a light-pink rectangle. The TSV area is calculated based on [81] and total number of pins for each of the Wide-I/O DRAM channels (6×44 with a pitch of 50 um). Considering the fact that, our explorations targets low-cost MPSoCs used in the embedded market, we do not use any kind of liquid cooling in our thermal model. As of the relative position of silicon dies in the 3d stacked structure, we have put the core die as the nearest one to the PCB and then four DRAMs dies are located on the top of the core die. At first glance, a better thermal balance in the chip may be obtained by putting the hottest die in the nearest position to the heat-sink. This idea however, is not practically feasible due to the very large number of supply related signals of the core die which should get connected to the suitable package pins.

For each thermal simulation, power trace for the core die, and separate power traces for each of DRAM dies are get calculated (as described in sections 4.2.2 and 4.2.1). The dimensions of the ARM Cortex-A9 CPUs are obtained from [31] and further scaled to 32 nm technology. For the DRAM controller units in the center of the core die, we use an averaged power value of a Wide-I/O DRAM controller reported by [14] (80 mW per controller). Practically, the core die contains more units than just CPU cores and DRAM controllers (such as GPU, WiFi, ...). We consider an averaged, typical power of 1.7 W (as reported by [56]) for all of these units and we divide this power evenly to the rest of vacant area in the core die.

We use a copper heat-sink with a heat transfer coefficient of $1.3 \times 10^{-9}\ W/(K.m^2)$ in our thermal model. This value is calculated based on [56, 101] and the chip dimensions.

4.2.4 Refresh Governor

At each simulation step the refresh governor receives the estimated temperatures and defines the refresh rates of DRAMs based on their maximum temperature (Table 4.1).

We perform statistical analysis on the temperature profile of our 3D MPSoC, and we measure lateral and vertical variation in temperature across different coordinates in the 3D structure.

For instance, with AndEBench, when all 8 CPU cores are running at 1.4 GHz, an averaged vertical temperature variation of 5.6 °C can be seen across 4 DRAM dies. In the first DRAM die, averaged lateral difference in temperature between two adjacent DRAM banks of a same channel is 3.3 °C. As Table 4.1 shows when the averaged DRAM die temperature is > 85 °C, the mentioned lateral and vertical temperature variations cause significant differences in the required refresh rate of each DRAM bank.

Due to these observations, we implemented the following key idea: instead of defining the refresh rate based on the maximum temperature seen across the entire channel and refreshing all DRAM banks at the same rate, we select the *refresh rate of each bank separately* based on its own maximum temperature.

As described in 4.2.1 we have extended our DRAM subsystem model to support handling of separate per-bank refresh commands. As we will show in section 4.3.1 this increases the overall refresh period (makes refreshes happen less frequently) and improves the performance of the system as well as the energy consumption.

4.2.5 Thermal Governor

Our thermal management solution is based on PID controllers [118]. Since the core die is always the hottest die in the chip, we focus on governing the temperature of the core die which results in controlled temperature over DRAM dies as well.

According to Figure 4.4, for each quarter of the core die, which has a defined threshold value (T_{TH}^{core}), we use a separate PID controller with its own error ($e = T_{TH}^{core} - T_{max}^{quarter}$) input. Each controller monitors the temperature of one MPCore unit. Required manipulation in the performance knobs is decided separately by each controller based on its error input and its state.

At each decision making step, for each performance knob, we evaluate the output of all controllers in each quarter and we select the maximum specified manipulation values. This guarantees the safety of the system operation.

Available performance knobs in our system are the clock frequencies of each MPCore and each DRAM channel. The supply voltage of each MPCore unit also changes according to its running frequency (Section 4.2.2).

Here, we focus on the operation and tuning of one controller. To be on the safe side, for each user-defined T_{TH} a lowered threshold T'_{TH} is created and used by the controller. This ensures that the user specified threshold will never get crossed. T'_{TH} depends on t_{sim}^{min} and thermal characteristics of the chip.

We estimate the maximum possible increase in the temperature of one on-chip block during one sampling interval and call it T_j °C/s. For each on-chip unit, we obtain T_j by applying a power pulse input to the chip and measuring the temperature response of each unit. Power pulse is created by running a power virus on all of the cores at the same time for a short time duration. Considering the fact that, static power changes non-linearly with temperature, to have an accurate estimation for T_j, before running the test, we first warm up the chip to a safe temperature region (≈ 70 °C). Having T_j and t_{sim}^{min}, T'_{TH} should be selected so that:

$$T_{TH} - T'_{TH} > T_j \times t_{sim}^{min} \tag{4.1}$$

The above equation is also the basis of selection of adaptive values of t_{sim} as shown in Table 4.2. In fact, with a maximum chip temperature of T_{max}, t_{sim} should always be selected so that $T_{max} + t_{sim} \times T_j < T_{TH}$. As a result of our experiments, the maximum T_j value for our 3D chip is below 7.0 °C/ms.

To obtain the proportional coefficient (K_P) of the PID controller, we conduct a calibration step. We set the K_P to the maximum clock of the CPUs. Thus each time the estimated temperature is higher than T'_{TH} by 1 °C, the core turns off until the next sample point. Running the power virus on the system, we decrease K_P step by step until the temperature gets near to the T'_{TH} (e.g. $T'_{TH} - T < 0.5$). At this point, we stop the calibration and K_P will be set to its previous step value. For the values of K_I and K_D we use Ziegler-Nicholas and Cohen-Coon tables [19, 118].

Table 4.2: t_{sim} vs. maximum chip temperature

Maximum Temperature	40	60	70	80	90	100
Sampling Interval (ms)	32	16	8	4	2	1

4.2.6 Sampling Governor

The t_{sim} parameter affects the speed of the simulation significantly. It determines how frequently the thermal profile of the chip should be estimated and governor routines should be invoked. Basically, a high resolution sampling in time is only needed when the temperature values for silicon dies are near critical [1] regions.

For each sampling interval, if T_{max} is the maximum current temperature of the entire 3D structure, and T_{TH} is the defined hard threshold of the thermal controller t_{sim} will be set to its smallest value when T_{max} approaches T_{TH} so that $T_{TH} - T_{max} < \sigma$. We quantify σ at section 4.2.5. t_{sim} grows gradually as the distance between T_{max} and T_{TH} increases.

In real-world applications, t_{sim} may be as small as the sampling interval of the thermal sensors of the chip. The speed of thermal sensor plays a crucial role to the accuracy of the thermal controller. Practical examples show sampling intervals of 0.1 ms and less for real thermal sensors [60, 112]. Without loss of generality, we select $min(t_{sim})$ equal to $t_{sim}^{min} = 1$ ms during our simulations. This sampling period is small enough to show the effectiveness of our thermal and energy closed loop management ideas completely.

Table 4.2 shows selected sampling intervals for each temperature range with an assumption of $T_{TH} = 95$ °C for the threshold of the thermal controller. For $T < 40$ °C, t_{sim} of 32.0 ms and for $80 < T_{max} < 90$ °C, t_{sim} of 2.0 ms are chosen. For temperature values above 90.0 °C, t_{sim} is 1 ms.

After each thermal simulation, the *sampling governor* obtains the new t_{sim}. It then updates the simulation interval in the TLM environment, as well as power and thermal models.

[1] e.g. temperatures in which the refresh interval of the DRAM should change frequently, or defined thresholds for the thermal controller units.

4.3 Experimental Results

We conduct a set of experiments to demonstrate the advantages of the previously described contributions. First, the temperature variation aware bank-wise refresh is presented. Then, the operation of thermal governor is evaluated. Finally, the speed-up gained by adaptive sampling is quantified.

4.3.1 Temperature Variation Aware Bank-wise Refresh

We execute two sets of simulations; In the first set the bank-wise refresh is disabled. For each DRAM channel, the refresh governor finds the maximum temperature of the channel. Then suitable refresh period will be selected (Table 4.1) and used for all DRAM banks of the channel. In the second set, the bank-wise refresh is active (Section 4.2.4). We perform the test while the thermal controller is off and all CPU cores are playing their own respective traces at 1.25 GHz. DRAMs are running at 200 MHz and ambient temperature (T_{amb}) is fixed at 45 °C (JEDEC standard [142]).

Figure 4.5 shows the refresh periods for 8 banks of the first DRAM channel. When the bank-wise refresh is off, refresh periods are equal for all 8 banks. When bank-wise refresh is active, the banks located on the lower dies have smaller refresh periods compared to colder banks at higher dies. For example, at $t = 20$ s all of the banks are refreshed with a period of 24 ms when the bank-wise refresh is off. However, when the bank-wise refresh is active, only the bank 0 and bank 3 are refreshed at this period and other banks are refreshed at higher periods. As we see in Table 4.3 the bank-wise refresh results in an average improvement of 24% in refresh rate, and 16.4% in averaged refresh power. In near future, half of the DRAM power will be related to refresh [91]. Thus, the proposed idea can significantly improve the total energy consumption.

4.3.2 Thermal Controller

We first quantify the impact of each of the mentioned performance knobs in Section 4.2.5 in controlling the temperature of the chip. We also measure the drop in performance while decreasing each of the DRAM and CPU clock frequencies.

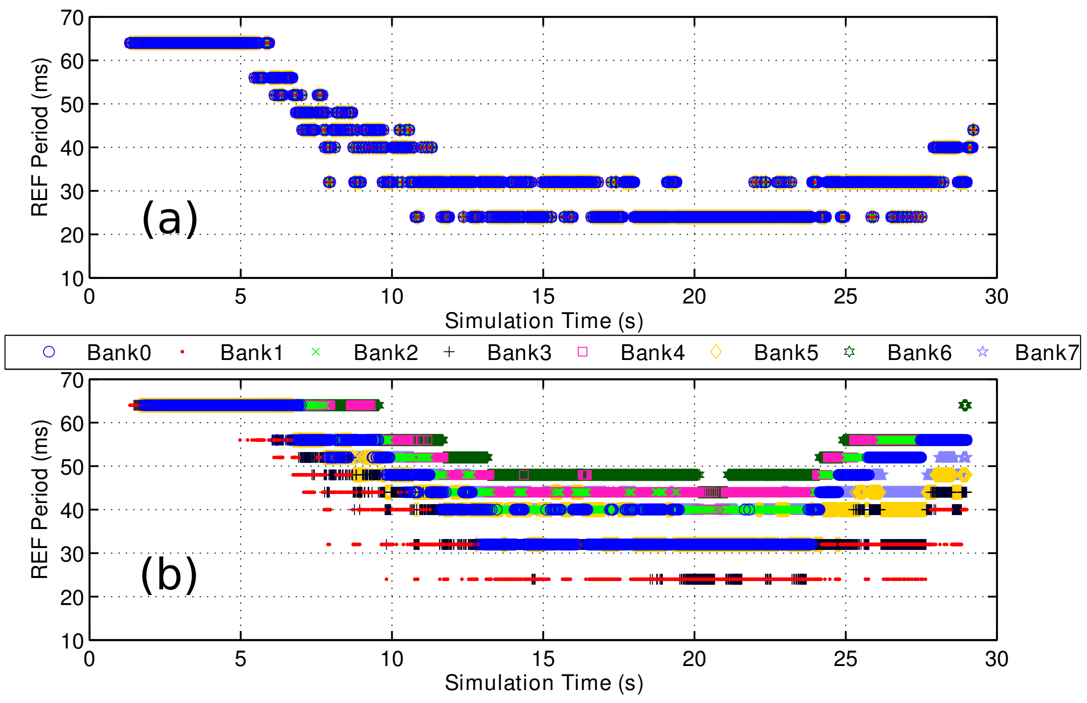


Figure 4.5: Refresh periods of Channel 3, (a) Bank-wise off (b) Bank-wise on (Smart-Bench).

For the DRAM clock we run the simulation and we decrease the knob from its highest value (200 MHz) towards the lowest (100 MHz). During these tests CPU clock stays constant at 1.25 GHz. We report the averaged temperature of the core and first memory die, as well as the total simulation time of the task and total consumed energy. We then keep the DRAM clock at its highest value (200 MHz) and scale down the CPU clock. Table 4.4 shows the results. Averaged execution time of each of the simulations in this table is 150 minutes. As we see, when DRAM is slowed down to 100 MHz, a decrease of 3 °C in DRAM die temperature will be obtained with 12.9% penalty in performance. However, by decreasing MPCore units clocks to 1050 MHz (and thus supply voltage to 1.0 V), the averaged temperature of first DRAM die will decrease by 8 °C while the performance drops only 11.3%.

As a result, for a desired decrease in temperature, scaling down the CPU clock gives better overall performance compared to DRAM. As a conclusion, in our controller we focus mainly on reducing the MPCore clock.

Table 4.3: bank-wise refresh vs normal method. Thermal controller off, CPU cores @1.25 GHz, All values are averaged.

Benchmark	AndEBench		0xBench		SmartBench	
Bank-wise Refresh	Off	On	Off	On	Off	On
Refresh Period (ms)	26.27	30.57	29.06	37.18	33.81	43.68
REF Power (W)	0.0297	0.0262	0.0221	0.0177	0.0181	0.0149
DRAM Power (W)	0.0842	0.0820	0.160	0.1586	0.1247	0.1232
IPC	0.5176	0.5236	0.5436	0.5515	0.4690	0.4733

Table 4.4: Impact of each control knob on the temperature and performance of the chip (Benchmark: smartbench)

Freq	Core T. (C)	MEM1 T. (C)	Sim. Time (s)	Energy (J)	Exec. Time(s)
Knob : DRAM Clock Frequency (MPCore @1.25 GHz)					
200	98.11	93.61	28.99	91.31	8915
166	96.16	91.88	30.86	91.65	8906
133	95.87	91.60	31.27	92.07	8936
100	94.74	90.60	32.73	92.62	9298
Knob : MPCore Clock Frequency (DRAM @200 MHz)					
1200	97.13	92.75	29.74	91.03	8970
1150	96.10	91.85	30.54	90.39	8956
1100	95.07	90.06	31.44	90.23	9046
1050	88.66	85.30	32.39	75.14	8661

We demonstrate the advantages of our thermal controller in terms of total consumed energy and the maximum temperature of the core and memory dies. For the set of benchmarks, we first run simulations with the thermal controller turned off and we record all the system statistics. Then we switch the thermal controller on and perform the same test. In these tests, we set T_{TH}^{core} to 100 °C. This indicates that our thermal controller should always keep the junction temperature below 100 °C.

Table 4.5 shows the results of each of the tests. The averaged execution time of each of the simulations in this table is 8 hours. As shown, in average, utilization of the temperature control mechanisms results in over 23% reduction in total energy,

Table 4.5: Key system statistics for each thermal controller test. ($K_P = 350$, $K_I = 1.17$, $K_D = 0.2$)

Benchmark	SmartBench		0xBench	
Test	Cntrl. off	Cntrl. on	Cntrl. off	Cntrl. on
Core die Ave. VDD (V)	1.1	0.98	1.1	1.04
Core die Ave. Frq (MHz)	1650.0	1181.8	1650.0	1374.3
DRAM Ave P. (W)	0.167	0.106	0.158	0.135
REF Ave P. (W)	0.0490	0.0118	0.0172	0.0110
Ave REF Period (ms)	20.97	53.83	41.58	56.63
Core+DRAM Ave P. (W)	4.66	2.83	3.81	2.96
Max Core Temp (C)	130.67	96.95	115.79	98.99
Max DRAM Temp (C)	119.5	91.86	102.82	91.80
Sim Time (s)	25.29	30.26	14.36	14.44
Total Energy (J)	117.85	85.63	54.71	42.74

25% decrease in DRAM power and more than 55% reduction in refresh power while the performance drops only by 10% and maximum temperature of the core die never passes the specified threshold.

To further stress the proposed thermal control policy and to show the flexibility of our infrastructure, we perform another test in which we run the *0xBench* benchmark and increase ambient temperature with a rate of 1 °C/s from 45 °C to 55 °C. Total simulation time is 14.68 and 15.14 seconds when the controller is off and on respectively. When the controller is off, the core die temperature increases up to 130 °C while with the controller active, the maximum seen temperature is 98.9 °C. The controller results in 37% improvement in overall system energy and 65% better refresh rate with a penalty of 17% in performance.

4.3.3 Adaptive Sampling

Figure 4.6 shows the dependency of t_{sim} to the die temperature for a test in which we turn off and on all CPU cores at $t = 10$ s and $t = 20$ s, respectively. The x axis is time, the left y axis is temperature and the right y axis is t_{sim}.

We quantify the speedup gained by adaptive sampling and ensure that the simula-

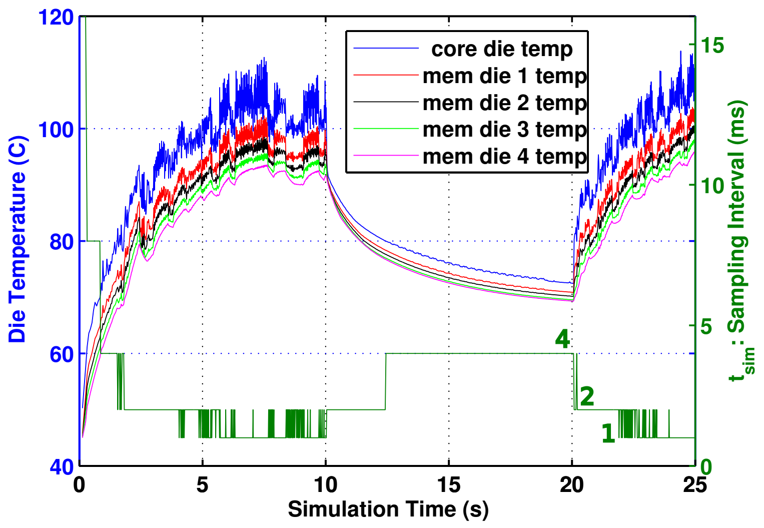


Figure 4.6: Change in sampling interval relative to temperature

tion accuracy remains at acceptable levels. We perform two tests: first, we execute the simulation task with the smallest fixed $t_{sim} = 1$ ms. Then, we re-run the same simulation and allow t_{sim} to change adaptively. The thermal governor is off during these tests and CPU cores are running *AndEBench* benchmark at 1.25 GHz.

For each test we record the total execution time of the simulation and complete history of temperatures. The accuracy of the proposed adaptive method is acceptable, if it reports similar *critical* temperature values at similar points in time as the fixed method. We refer to the fixed trace containing temperatures of the core die and for each point at time with temperature $T > T_{crit}$ we look-up the temperature of the identical point in adaptive method and calculate the difference.

The execution time of the simulation is 1.46×10^4 and 6.6×10^3 seconds for fixed and adaptive methods respectively. This is a speedup of $2.21X$. Averaged t_{sim} for adaptive method is 1.99 ms. For $T_{crit} = 90$ °C, the maximum temperature difference between identical points in adaptive and fixed traces is 0.87 °C.

4.4 Hardware Acceleration of Thermal Simulation

Thermal simulation is very computational intensive. Based on our practical measurements, more than 65% (with adaptive sampling) to 90% (fixed sampling) of the simulation time is dedicated to thermal simulation and its related sub-routines.

We study the feasibility of accelerating the execution of 3D-ICE with the aid of two different approaches:

1. Running one thermal simulation task on multiple CPU cores in parallel.

2. Hardware acceleration of thermal simulation using a specialized hardware.

Using the OProfile [86] performance analyzer, we first identify the *execution time hotspots* of the 3D-ICE. SuperLU [38] routines used for L-U factorization of matrices and solving systems of linear equations are the most computational intensive parts. SuperLU is built on the top of CBLAS [13]. Our investigations using OProfile show that the tool is running the CBLAS DGEMV (matrix-vector multiplication) function during more than 90% of its execution time.

First, we accelerated 3D-ICE by linking CBLAS against the Intel Math Kernel Library (MKL) [75]. This enables the tool to run computational intensive matrix calculation tasks on multiple CPU cores concurrently. We obtained a thermal simulation speedup of $> 2.1X$ on an Intel Core i7-860 CPU when utilizing 4 cores in parallel using MKL.

Second, we select the Maxeler [96] hardware acceleration engine (featuring one Xilinx Virtex6-SX475T FPGA) as target platform and adapt the 3D-ICE source code to the Maxeler development flow. This allows us to off-load any computational intensive part of the software to the hardware containing our computational kernels.

To obtain a measure of feasible level of speedup by the Maxeler hardware, we focused on hardware acceleration of the key routine: *DGEMV*. We implemented the computational kernel using *maxj* [96] and validated its functionality in practice. A speedup of more than $12X$ is seen for execution of DGEMV in comparison with one core of Intel i7-860 (at 2.8 GHz) while the FPGA is running at 150 MHz and 75% of its hardware multipliers and less than half of its logic slices are consumed.

Considering the fact that DGEMV is the main computational element of the SuperLU subroutines, similar speedups are also possible if we efficiently port the whole subroutine into the hardware. The detailed implementation of SuperLU routines for the Maxeler flow is more involved and is considered as a future work.

Chapter 5

Energy and Performance Evaluation of CPU-Accelerator Memory Sharing Using Xilinx ZYNQ

In the previous chapter, we studied the effect of considering for temperature variation on optimizing energy consumption in Wide-I/O DRAM. We used a TLM model for the DRAM to reach our goal. To continue research on temeprature variation aware energy optimization in MPSoCs, we need to focus on platforms with the same level of complexity as real-world ones. For such large platforms however, building TLM models even for the entire platform can never provide us with acceptable level of speed and accuracy.

As a consequence, we build multi-core heterogeneous system-on-chips using real-world configurable FPGA chips. We use the Xilinx ZYNQ [140] all-programmable heterogeneous SoC (APSoC) as our development platform. In the first part of this Chapter, we provide an overview on the ZYNQ architecture.

Considering the significant impact of specialized hardware accelerator units on the performance and energy consumption of modern MPSoCs, we also study efficient methods of utilizing hardware accelerators in a multi-core hardware architecture. Towards this, we focus on the machanisms through which the CPU and accelerator share the data and address values for processing tasks [6]. Through a set of practical experiments, we quantify the performance and energy efficiency of each of the provided interfaces in the ZYNQ device for sharing of data between hardware accelerator and the host.

In Chapter 6, we build a complete infrastructure featuring clusters of OpenRISC cores on the ZYNQ device. This infrastructure can be used for our future research regarding efficient methods of address sharing between the CPU and accelerator as well as the ideas related to temperature variation aware hardware acceleration.

5.1 Related Research Works

Efficient sharing of data among computational units in a system plays a strategic role in performance and energy optimization.

Heterogeneous System Architecture (HSA) foundation provides architectural and application level solutions to help system designers integrate different kinds of heterogeneous computing units in a way that eliminates the inefficiencies of sharing data and sending work items between them [83].

The developed acceleration hardware blocks become HSA compliant by declaration of the necessary low-level interface layers. This frees the programmers from the burden of tailoring a program to a specific hardware platform. As a part of HSA, [107] describes *AMD Fusion System Architecture*; targeted to unify CPUs and GPUs in a flexible computing fabric. This allows the accelerator logic located on the GPU to use the virtual address values passed by the CPU directly without the need for conversion to equivalent physical address. The proposed idea however, is mainly developed for sharing memory between CPU and fully programmable GPU cores and is not targeting reconfigurable heterogeneous architectures like *ZYNQ*.

A methodology for analyzing the impact of hardware accelerator data transfer granularity on the performance of a typical embedded system is presented in [84]. This is particularly important because, as we will show, the granularity of data transfer between memory and accelerator, and thus, the interrupt rate to the CPU has direct impact on the performance of the system.

The idea of using a portion of CPU sub-system caches as buffers for the accelerators is studied in [44]. This results in smaller silicon area since each accelerator doesn't instantiate its own buffer. The basic idea of dedicating a shared memory space to accelerators is interesting because the *ZYNQ* device provides a dedicated On-Chip

Memory (*OCM*) which can be used for the same purpose. We also consider processor-accelerator memory sharing using *OCM* in our tests.

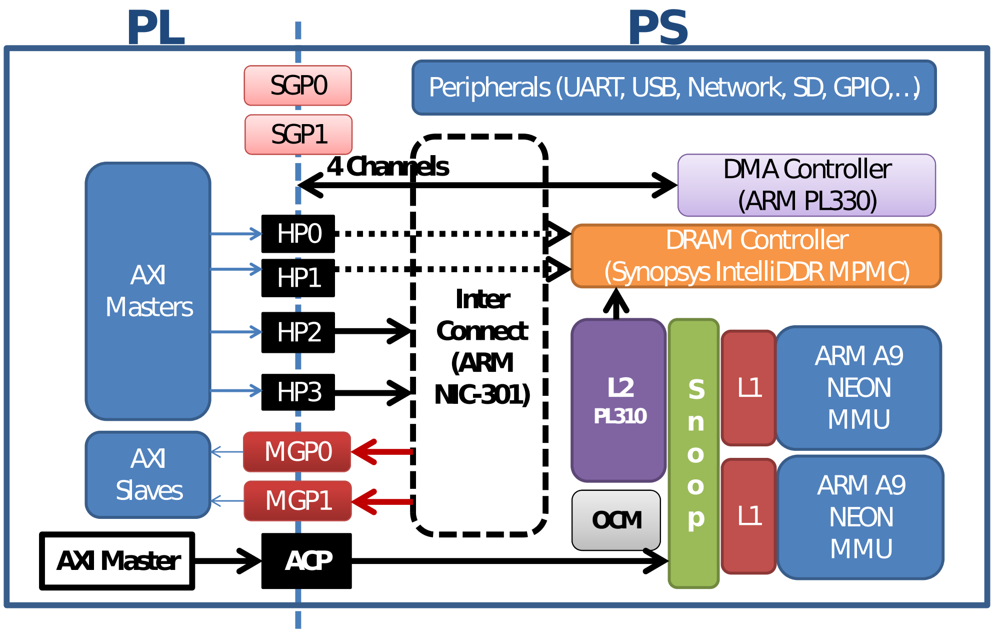
The problem of maintaining coherency between CPU caches and accelerator data in a multi-core embedded system is addressed in [20]. The paper discusses possible hardware architectures and related software solutions to tackle the problem. It concludes that the optimal solution heavily depends on the characteristics of the application. The paper discusses the solutions at architecture level and does not provide detailed practical comparisons on the performance and energy efficiency of each solution.

An area- and power-efficient many-core computing fabric which features clusters of up to 16 processor cores is proposed in [18]. The developed platform delivers an extraordinary level of computational speed ($>$ 80 GOPS) while consuming relatively small amount of power ($<$ 2 W). The paper is of particular importance since it provides ideas on the development of energy efficient accelerator logic. Indeed, the developed architecture in our paper is partially inspired from [18].

A high-performance, energy improved mobile processing platform named big.-LITTLE is introduced by [53]. The platform consists of high performance Cortex-A15 processor and energy efficient Cortex-A7. The connection between the CPU subsystems is provided through the CCI-400 interconnect which facilitates full coherency between Cortex-A15 and Cortex-A7 as well as GPUs, accelerators and I/O. This platform, if connected to a programmable gate array, can provide a suitable testbed for evaluation of various processor-accelerator memory sharing schemes.

The impact of cache architecture on the performance and area of FPGA based processor and parallel accelerator systems is discussed in [32]. The paper proposes a simple hardware containing one MIPS core, multiple accelerator units, a multi-port shared L1 cache and a DRAM controller. It considers different structural parameters for the L1 cache (such as number of ports, associativity, etc.) and defines a set of computational tasks to be done only by accelerators. It then quantifies the impact of cache structure on the overall speed of accelerators connected to the L1 cache. The paper does not discuss the cooperative operation of CPU and accelerators. Moreover, the developed hardware in the paper is very simple. It is not capable of booting an operating system and communicating with the outside world.

The idea of adding hardware accelerators to reduce power in FPGAs is investigated in [64]. The paper shows practical comparisons for the power consumption of sample



PL
PS
SGP0
SGP1
Peripherals (UART, USB, Network, SD, GPIO,...)
4 Channels
DMA Controller (ARM PL330)
AXI Masters
HP0
HP1
HP2
HP3
DRAM Controller (Synopsys IntelliDDR MPMC)
Inter Connect (ARM NIC-301)
L2 PL310
Snoop
L1
ARM A9 NEON MMU
L1
ARM A9 NEON MMU
AXI Slaves
MGP0
MGP1
OCM
AXI Master
ACP

Figure 5.1: A block diagram representing important elements of the Xilinx ZYNQ device.

computational tasks, when they are executed by the CPU or the accelerator logic. The paper does not address issues related to coherency and processor-accelerator memory sharing.

To the authors knowledge, our work is the first one which practically quantifies the potential processing bandwidth and energy efficiency of different processor-accelerator memory sharing methods using the *ZYNQ* device.

Moreover, it is the first work which provides an explicit practical comparison in terms of energy and speed, on processor-accelerator memory sharing using *ACP* and other traditional methods. In addition, we also provide a flexible research vehicle which facilitates evaluation of innovative ideas regarding the design of hardware accelerators in heterogeneous architectures on programmable gate arrays.

5.2 Xilinx ZYNQ Heterogeneous System-on-Chip

Xilinx ZYNQ device [140] contains two parts:

- Programmable Logic *(PL)* which is roughly a full FPGA.

- Programmable System *(PS)* which is a complete sub-system with ARM CPU cores and different peripherals.

The PS contains the following items:

- ARM MPCore-A9 dual core processing engines which also contain NEON SIMD units.
- Each ARM core has its own $L1$ data and instruction caches. Each cache block has a size of $32KBytes$.
- One $L2$ cache with the size of $512KBytes$ which is shared between two CPU cores. The ARM *PL310* cache controller is used for implementation of this unit.
- A Snoop Control Unit (SCU) which ensures coherency between the contents of the caches.
- An on-chip memory (OCM) which is a multi-port memory block of $256KBytes$ and can be accessed by the CPU or other ports and units in the system.
- A DMA controller which can be used for transferring data between peripheral and DRAM memories. It also provides 4 DMA channels to logic residing on the PL side of ZYNQ. This unit is based on ARM *PL330* PrimeCell.
- A multi-port memory controller, which is responsible for connecting to DRAM memories and receiving reqd/write requests from different sections of the hardware and passing them to DRAM. This block is mainly a design of Synopsys.
- A large ensemble of different peripheral such as UART, Gigabit ethernet, USB peripheral and host, CAN bus, I2C Bus, SD Card interface, I2C interface, GPIO and so on..., which can be configured and used by the ARM CPU cores very easily.
- An interconnect based on ARM *NIC-301* design which connects differnt blocks of hardware inside PS together.
- Finally we have a set of AXI interfaces (as shown in Figure 5.1) are implemented to make the communication between PS and the PL logic possible.

Basically these AXI interfaces divide into two groups:

- AXI Master interfaces (*GP*), connect to AXI slaves residing on the PL. The CPU is able to initiate read/write transactions over these AXI masters to transfer data to PL modules. There are two 32 Bits AXI master ports available in the ZYNQ device: *GP0* and *GP1*.

- AXI Slave interfaces (*HP*, *ACP* and *SGP*), connect to the implemented AXI masters on the PL. There exist four *High Performance* (HP) ports and one *Accelerator Coherency Port* (ACP). Each of these interfaces implements a full-duplex 64 Bits connection, meaning that at every clock cycle, total 16 Bytes of data can be transferred on *AXI read* and *AXI write* channels concurrently. The two *SGP0* and *SGP1* interfaces implement 32 Bits connections.

There exists a defined memory map for the *ZYNQ* device [140] which indicates the address range of each logic block. Every AXI slave unit, implemented on the PL will also occupy a part of this address range. It should be noted that except the CPU cores and their $L1$ instruction caches, the rest of the system is using physical address values. The *HP* and *ACP* ports have both 64 Bits width.

The *ACP* port is connected to the ARM Snoop Control Unit (*SCU*). Thus it provides the possibility of initiating cache coherent accesses to the ARM sub-system by the master on *ACP*. Careful use of *ACP* can improve overall system performance and energy efficiency. Inappropriate usage of this port however, can adversely affect execution speed of other running applications because the accelerator can pollute precious cache area.

5.3 Infrastructure for Energy and Performance Quantification of CPU-Accelerator Data Sharing

We develop a complete infrastructure containing hardware, software and firmware setup which enables us to perform evaluations on different processor-accelerator memory sharing methods. For the firmware, we basically use the generated firmware by Xilinx tool-set for our target board (*ZC-702*). We ensure that AXI level shifters are enabled from the beginning of device operation. This is vital for the correct operation of *HP* and *ACP* interfaces.

5.3.1 Hardware

Figure 5.2 shows a block diagram of the developed hardware on the ZYNQ device. As we see, three AXI slave interfaces (*ACP*, *HP0* and *HP1*) and one AXI master interface (*GP0*) are enabled and used.

The AXI masters used in this design implement *AXI 4.0* protocol specifications [68]. They are based on the AXI master template provided as a LogiCORE by Xilinx [71]. Each AXI master logic is further customized to issue an interrupt when it finishes a transfer task. The interrupt signals are connected to the interrupt controller unit on the PS. Each AXI master, also contains an AXI slave port, through which the CPU can program and start the master. All of the AXI masters are configured to handle burst lengths of up to 256 which is equal to 4096 Bytes of data (*read+write*). Each AXI master, when programmed, is capable of handling transactions of up to 1 MBytes total length. The AXI slave side of all of the AXI master units residing on *ACP*, *HP0* and *HP1* ports are connected to *GP0*.

In order to push the processing speed to its maximum, we use two AXI master blocks (called *AXI read* and *AXI write*) running in parallel to perform concurrent read and write transactions on each PS interface. The masters are connected to the interface using an AXI interconnect. In Figure 5.2 each AXI interconnect is depicted with a big *i* letter. AXI interconnects are configured to their high performance cross-bar mode to achieve maximum possible bandwidth.

As shown in Figure 5.2, between *AXI read* and *AXI write* there is a separate module (called *acceleration logic*) which contains a FIFO and also the logic related to the acceleration task. For our performance measurements, we have selected a 16 tap FIR filter as the acceleration logic.

Each pair of AXI masters can operate in parallel, meaning that, while one of them is reading a packet of data from the memory, the other one is writing the previous processed packet back to its destination. The developed driver at software side is responsible for synchronization of these two units.

The proposed hardware provides the designers with an architecture which is very

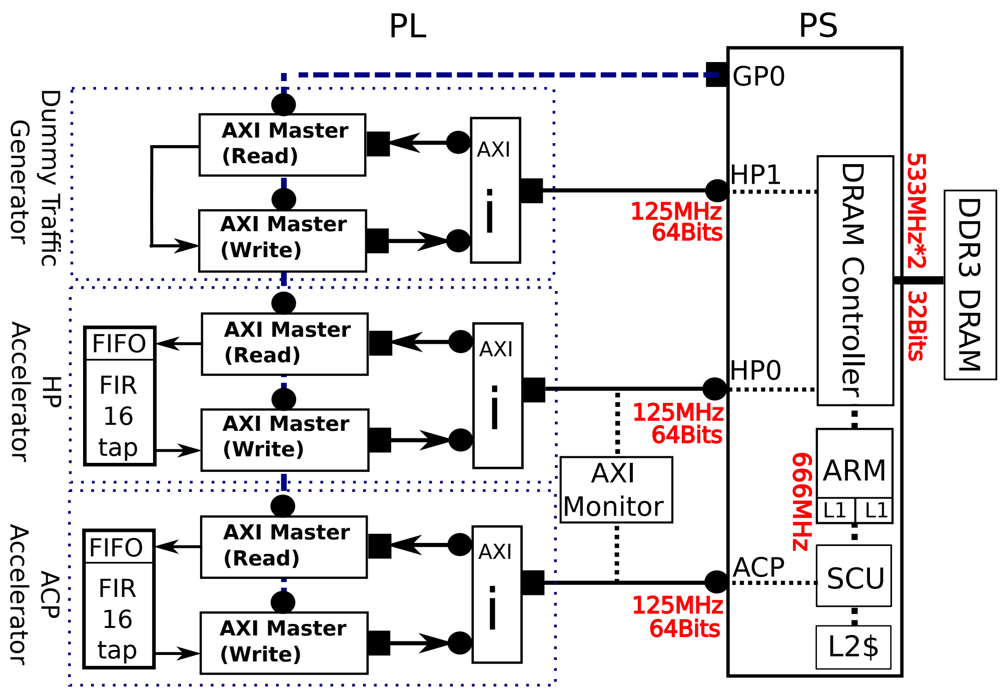


Figure 5.2: Block diagram of the developed hardware.

easy to customize. For any defined computational task, the *acceleration logic*, which is based on an easy to understand FIFO interface, is the only block which needs to be modified.

The hardware also includes an AXI monitor unit [70]. Its purpose is to be able to monitor the AXI interface signals, so that we can debug possible problems and further investigate latency values by directly looking at the actual waveforms.

If the AXI interfaces are running at 125 MHz the maximum *theoretical* full-duplex bandwidth for them is 2 GBytes/s. For DRAM memory, the data bus width is 32 Bits and if *DDR3 DRAM* is running at 533 MHz, we reach a *theoretical* bandwidth of 4.2 GBytes/s.

Finally, we have also placed a set of AXI masters on the *HP1* port. The purpose of these blocks is to be able to generate *dummy* traffic to the DRAM whenever required. Using this block we will evaluate the performance of the accelerator running on *HP0* while the DRAM is also busy handling other incoming requests.

We have implemented this hardware on the ZYNQ *XC7Z020-1C* device available on Xilinx *ZC-702* board. The total number of used logic slices is equal to 7324 which corresponds to 55% of the total available slices. The design consumes 92 block memories of 36 Kbits size (65%) and 9 block memories of 18 KBits size (3%). The maximum clock frequency for this design is 128.2 MHz.

5.3.2 Software

Our developed software is divided into two major parts: Linux kernel level drivers and user level applications. At Linux kernel level, our infrastructure consists of two drivers which are called *axiD* and *axiD dummy generator*.

axiD manages the AXI masters located on *HP0* and *ACP* ports. The driver is responsible for:

- Memory allocation and obtaining the physical address of allocated memory buffers which will be used by AXI masters. Memory can be allocated in either cachable or non-cachable regions depending on the memory sharing method. (Further descriptions in section 5.3.4.)

- Initializing, programming and triggering the AXI masters and handling the interrupts generated by them.

- Calculating the source and destination addresses for the set of accelerators based on the status of the ongoing processing tasks, number of passed loops and number of processed data chunks.

- Interaction with user-level applications: receiving raw input data from user side, copying to source memory buffers and then writing back the processed results to user-level.

- Providing an accurate tool to measure time intervals. The driver enables access to the free running 64 Bits counters of the ZYNQ device [140] which are clocked at 333 MHz (half CPU clock frequency).

- Configuring the *PL*310 [67] cache controller statistics unit so that it reflects total number of read requests received at the cache and the total number of read hits. The driver reflects these values at the beginning and ending time of each task.

The developed *axiD dummy generator*, does not perform any acceleration related task. When needed, it enables us to activate the dummy traffic generator AXI masters residing on *HP1* port.

At user level, we have prepared the following items:

- A simple application which communicates with the *axiD* driver.
- A simple memory intensive application (called *background application*), which allocates a memory buffer and performs arbitrary read and writes to this buffer in an endless loop. This application will be used to demonstrate the effect of cache pollution on the performance of *ACP* accelerator.
- The *Oprofile* statistical performance monitoring tool [86] which we have ported to the ZYNQ environment. This enables us to measure important performance metrics of the CPU sub-system.

5.3.3 Power Measurement

Since we are interested in accurate quantification of the consumed energy of the acceleration methods (disscussed in 5.3.4), we need a power measurement solution with enough resolution in measured values and also in time.

The *ZC-702* board is utilizing a set of power supply units which provide online sensors for voltage, current and temperature measurement [72] [1]. They are connected to the outside world through the *PMBus*. Using the *TI USB Interface Adapter PMBus pod* and the *TI Fusion Digital Power Designer* software we can monitor and log the voltage and current values for different sections of the ZYNQ board.

Basically we sample the following consumed power values:

1. Core logic for the *PL* part of *ZYNQ*.
2. Internal logic of the *PS*.
3. Interfaces and I/O buffers of the *PS*
4. ob-board DRAM chips.

[1]This is also true for the ZC-706 board which we for implementation of our OpenRISC cluster described in Section 6.1.

Based on our practical observations, these four items are the most power hungry parts of the system. The sampling frequency for measurement of voltage and current values is equal to 2 Hz.

5.3.4 CPU-Accelerator Data Sharing Methods

In order to evaluate different processor-accelerator memory sharing methods in terms of speed and energy efficiency, we first define a processing task. Then we define a set of processing methods to accomplish this task. Each processing method utilizes a different memory sharing scheme. We then execute each processing method on the real hardware and measure performance and power.

Processing Task: For a sample image of *i* bytes, perform the following: read the image from the source buffer, pass the image through the FIR filter, and finally write the output back to the destination buffer. Source and destination buffers are different. In practice, we continuously perform this operation a large number of times. This enables us to have an accurate speed and power measurement.

Data Sharing Methods: Here, we describe the methods that we use to perform the processing task. We assign a name to each method, which will be used during the rest of this Section.

- *HP0 Only*: The accelerator located on *HP0* is responsible to perform the processing task alone. Image source and destination buffers are allocated on the DRAM memory and in the non-cachable area. (Linux kernel call *dma_alloc_coherent* is used for this purpose.)

- *ACP Only*: The accelerator located on *ACP* is responsible to accomplish the processing task alone. Image source and destination buffers are allocated using normal *kmalloc* Linux kernel call, thus, they are allowed to also be cached by CPU sub-system.

- *OCM Only*: The accelerator located on *ACP* is responsible to accomplish the processing task alone. However, image source and destination buffers are located in the On-Chip Memory (*OCM*) block of the ZYNQ device. Here the allocation will be done like other hardware peripherals using *request_mem_region* and then *ioremap* Linux kernel calls.

- *CPU Cache*: The CPU core is responsible for doing the processing task alone. No accelerator is active. The source and destination image buffers are allocated using *kmalloc* thus, they are allowed to get cached.
- *CPU no Cache*: Is similar to *CPU Cache* however, memory allocation for the source and destination buffers is done using *dma_alloc_coherent* thus they are located on non-cachable region of memory.
- *CPU HP0*: The CPU and the accelerator on *HP0* port cooperate to perform the processing task. At each iteration first the CPU reads the source image, performs the processing and writes the result back to the memory. Then it is the turn of the accelerator on *HP* port to perform the processing task. Image source and destination buffers are allocated on non-cachable region of memory.
- *CPU ACP*: Is similar to *CPU HP0* however, the accelerator on ACP cooperates with CPU to accomplish the task and image buffers are allowed to be cached.
- *CPU OCM*: Is similar to *CPU ACP* however, source and destination image buffers are located on the *OCM*.

5.3.5 Experimental Results

We consider the processing task described in section 5.3.4 and we use each of the described methods to accomplish this task and to measure processing speed and energy.

We sweep over different image size i values to evaluate the effect of used memory size on the speed of operation. ($i = \{4, 16, 64, 128, 256, 1024, 2048\}$ KBytes). By increasing i, we also increase the size of packets (p) transferred by the AXI masters ($p = \{4, 16, 64, 128, 128, 128, 128\}$ KBytes). Although our AXI masters are capable of handling packets of up to 1 MBytes, we limit the packet size to 128 KB which is the size of FIFOs inside *acceleration logic*. During these tests, we use a fixed running frequency of 125 MHz for the entire logic residing on the *PL*.

We measure total execution time and thus total processing bandwidth (*read*+*write*) for each case. During each test we also measure total number of $L2$ cache requests and hits to have a better insight on $L2$ cache utilization.

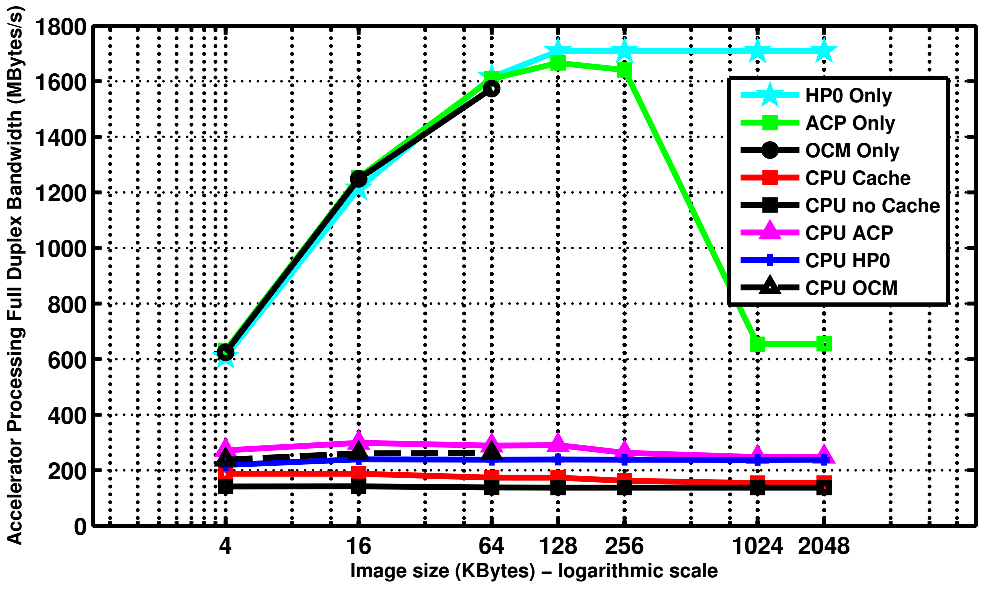


Figure 5.3: Processing bandwidth comparison of acceleration methods. Image size sweeps from 4 KB to 2048 KB.

Figure 5.3 shows the total processing bandwidth for each method. The Y axis represents total transferred data ($read + write$) in MB/s. X axis represents the size of image (i) being processed in a logarithmic scale.

The following processing methods show highest performance: *HP0 Only*, *ACP Only* and *OCM Only*. For *OCM Only* we can perform the processing task only for limited values of i since, the total On-chip Memory available on the *ZYNQ* device is limited to 256 KB. For $i = \{4, 16, 64\}$ KB we see almost equal performance for each of these three methods. At $i = 128$ KB and $i = 256$ KB, we notice a slight decrease in the performance of *ACP Only* compared to *HP0 Only* (1708.5 MB/s for *HP0 Only* vs. 1665.9 MB/s and 1640.8 MB/s for *ACP Only*). When image size grows over 256 KB, a significant drop appears in the performance of *ACP Only* (653.3 MB/s for *ACP Only* vs. 1708.5 MB/s for *HP0 Only*).

This phenomena can be described as follows; For each processing task, the total utilized memory (by source and destination image arrays) is $2 \times i$. If the total available cache size is L, then while $2 \times i < L$ the system is able to efficiently store local copies of recently used *ACP* accelerator data on its caches, thus providing fast access to data when needed. However when $2 \times i > L$, it is no more possible to cache the entire data objects used by the accelerator. As a result, some accelerator requests to the cache will fail and will eventually end-up the DRAM memory. The extra delay introduced by passing through the caches to DRAM, causes a serious decrease in performance. Either an increase in i (e.g. increasing the size of processed image) or a decrease in L (e.g. a background application is also consuming available caches) can cause the above phenomena. In Figure 5.3, no background application is running on the system. The size of available shared $L2$ cache is 512 KB in the ZYNQ device. As we see, performance drop happens when $2 \times i > 512$ KB.

We now consider processing methods which fully or partially use the CPU cores to perform the processing task. *CPU no cache* method is showing the lowest performance (average 140 MB/s) and *CPU cache* is slightly higher (average 170 MB/s). Here, the entire processing is done only with the ALU of the CPU. Enhancements in speed is possible if we use the *NEON SIMD* engine of ARM CPU cores. But even in that case the possible speed-up is around $8X$ [65].

Finally, we have *CPU ACP*, *CPU OCM* and *CPU HP0* with speeds between CPU only and hardware only methods. Here, cooperation of accelerator with the CPU causes an speed-up in data processing. *CPU ACP* is always faster than *CPU HP0*. This is because of the possibility of sharing the data between CPU and accelerator on the cache (Thus the CPU can access data faster). The speed of *CPU OCM* is always between the other two methods. For $i \leq 256$ KB *CPU ACP* is approximately $1.22X$ faster than *CPU HP0*. By growing the image size however, the speed of *CPU ACP* begins converging to *CPU HP*.

Looking at the number of $L2$ cache hits, we notice a significant difference between the methods which use *ACP* and methods which use *HP0*. For example, in *ACP Only* we see more than 2 hits per each 32 bytes (one cache-line) of processed data while for *HP0 Only* this value is practically zero (in the order of 10^{-5}).

For each test point in Figure 5.3, we also measure power. Figure 5.4 shows the results. Considering the fact that, power values do not change significantly by changing the image size for each processing method, we only show the averaged power value for all of the image sizes.

In Figure 5.4 we show the four major power sinks (as described in section 5.3.3) of the *ZC-702* board at the time of the tests.

As shown, *HP0 Only* method has the highest DRAM power. *CPU cache* causes

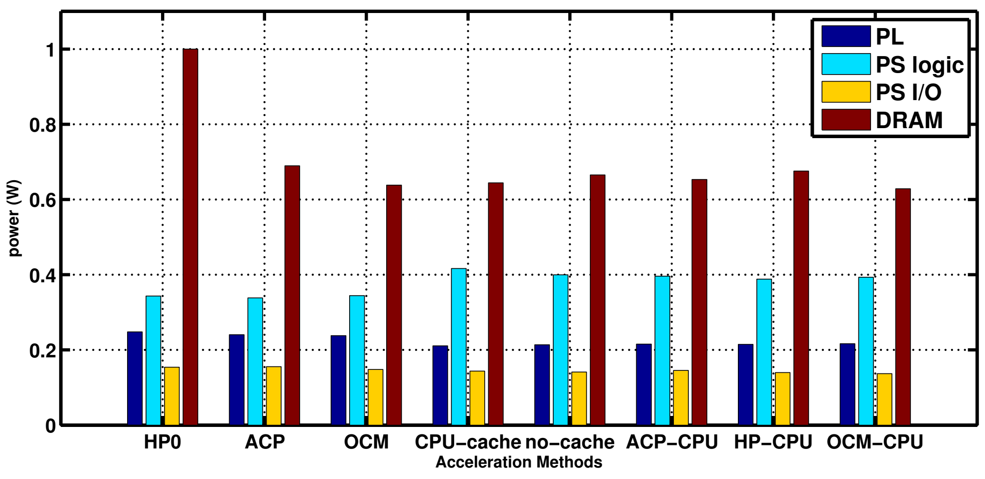


Figure 5.4: Averaged power consumption of major system blocks for each processing method.

highest power consumption by PS internal logic, and *HP0 Only*, *ACP Only* and *OCM Only* show highest values of power consumption by the PL. Power consumption of *PS* I/O buffers are at the same level for all methods.

Having the processing bandwidth and power, we calculate the energy consumed for processing one byte of data. Figure 5.5 shows energy values for each of the test points.

Looking at Figure 5.5 we see, for $i \leq 256$ KB, *ACP Only* and *OCM Only* consume the least energy. For $i > 256$ KB however, *HP0 Only* shows better results. At the next level, among methods which utilize the CPU, cooperation of CPU and accelerator over ACP (*CPU ACP*) shows the lowest energy. After that, we have *CPU OCM* and then *CPU HP0* showing more energy consumption.

Effect of Background Workloads: Now, we study the effect of background workloads on the performance of *ACP Only* and *HP0 Only* methods. First, we turn on the dummy traffic generator on *HP1* AXI interface. This block continuously performs arbitrary read and write operations to an allocated 2 MB DRAM area. At the same time we measure the performance of *ACP Only* and *HP0 Only* for different values of i.

In another test, we execute a memory intensive *background application* on ARM CPU cores. The dummy AXI traffic generator is off. The *background application* performs arbitrary read and write operations to an allocated 2 MB array. The array is

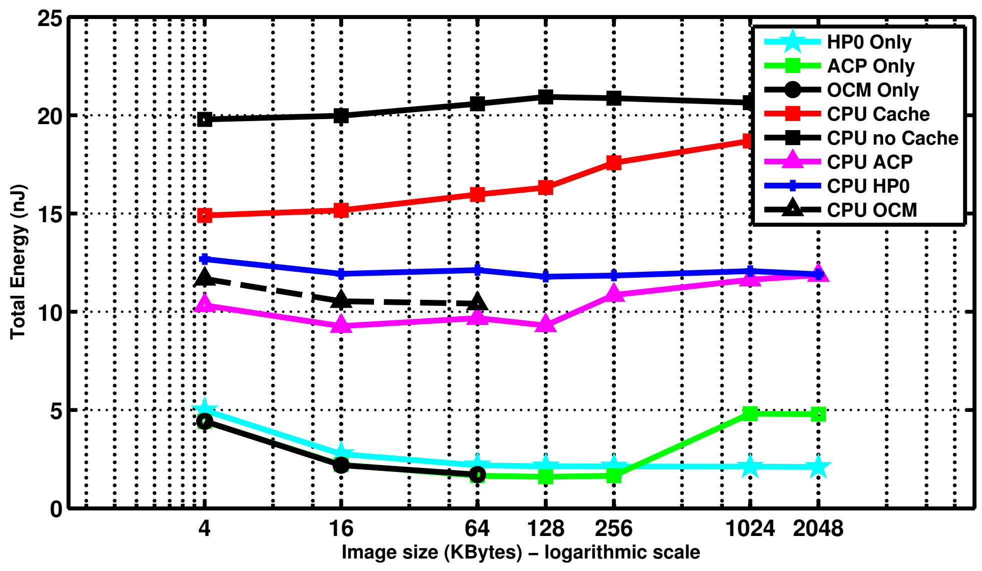


Figure 5.5: Energy consumed for processing of one byte of data for each processing method.

allowed to be cached. Thus, it occupies CPU caches during execution. In fact, this test shows the effect of decreasing L on the performance of the acceleration method.

Figure 5.6 shows the results of both tests. In this figure, X axis is i and Y axis is total transferred data in MBytes/s. We first look at the effect of *AXI dummy* activity on performance.

As we see, performance drop of (*ACP Only*) is negligible. However, a major drop can be seen in the performance of *HP0 Only*. For example, at $i = 128$ KB, *HP0 Only* speed is 1708.5 MB/s when there is no other activity going on DRAM. However, its speed drops to 1382.2 MB/s when the *AXI dummy* interface is active and occupying DRAM bandwidth. For *ACP Only* the corresponding numbers are 1665.9 MB/s and 1664.3 MB/s respectively.

Looking at the results of the second test where the cache is heavily occupied by the *background application*, we see a clear drop in the performance of *ACP Only* while *HP0 Only* shows a slight performance shift. In *ACP + background application*, the speed of the ACP accelerator does not grow for $i > 16$ KB. For example at $i = 128$ KB, the speed of *ACP Only* is 1665.9 MB/s and 531.6 MB/s with and without the *background application* running, respectively.

Another noticeable point during the second test is that: the operation speed of both

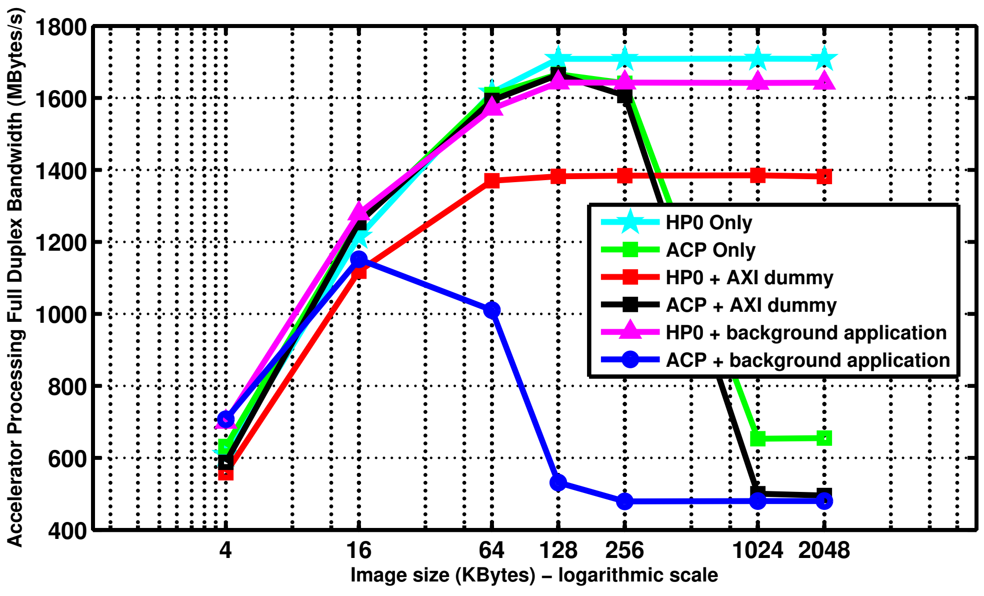


Figure 5.6: Processing bandwidth comparison of ACP and HP0 accelerators at the presence of background traffic.

ACP and *HP0* accelerators, at $i = 4$ KB, is higher when the *background application* is running on the CPUs compared to when the CPUs are not doing any specific task [1].

We describe this phenomena as follows: when the background application is running on CPU cores, it keeps the CPU sub-system active preventing it to go to idle state. Thus when an AXI master finishes its current task and issues an interrupt request to the CPU, the service routine will get executed in a shorter time, and the master begins the next task faster. This description can be further confirmed by noting the fact that when the packet size increases (and thus the rate of AXI master interrupts to the CPU decreases) this speed-up disappears.

[1]700.0 MB/s for *HP0* and 707.3 MB/s for *ACP* when *background application* is running compared to 608.5 MB/s for *HP0* and 631.8 MB/s for *ACP* while the CPUs are idle.

Chapter 6

Temperature Variation Aware Hardware Acceleration in Heterogeneous MPSoCs

In this Chapter, we mainly emphasize on the future directions of our research. We study the idea of accounting for on-die temperature non-uniformities while scheduling computational tasks to hardware accelerator units.

First, we design and practically create a high performance computation architecture which can be efficiently used towards our future research on temperature variation aware hardware acceleration. This architecture also enables us to conduct research regarding efficient sharing of data between host CPU and accelerator units in a heterogeneous Multi-core SoCs. Section 6.1 illustrates the developed hardware infrastructure. Section 6.2 describes our on-going and future research in more detail.

6.1 OpenRISC Based Heterogeneous Multi-Core SoC

We now build a complete architecture designed to be capable of accelerating applications developed by OpenCL [80] and OpenMP [110]. This allows for execution of a wide range of real-life benchmarks on our platform. Our infrastructure contains two clusters of fully functional 32 Bits OpenRISC [109] CPU cores as the acceleration logic. Each cluster contains 4 OpenRISC cores. In addition, each cluster features

512 KBytes of data memory. The data memory banks are acting as *L1* memory and are tightly coupled with OpenRISC cores, so throughout this text we name these memories as Tightly Coupled Data Memories (TCDM). A logarithmic interconnect unit provides the required connections between OpenRISC cores and each of these memory banks. This architecture allows for efficient sharing of data between OpenRISC cores as well as a high level of computational performance.

6.1.1 Hardware Architecture

In order to facilitate the understanding of this design, we show the implemented architecture using 3 figures. First we begin from the top-level block diagram. Then we go towards the lower levels of the design.

Figure 6.1 shows a simplified top-level block diagram of the design. In this figure, the two instances of OpenRISC clusters are indicated as *Cluster Modules*. The AXI interconnectes are denoted with *i* letterns. The *C* blocks are responsible for keeping the address bus width in the necessary range. Their operation will be further described in Section 6.1.2. Throughout this text we call the ensemble of *Cluster Modules*, *L2* memory, *RAB Table* and their connected AXI interconnects as *accelerator* or *cluster system*.

As shown, each cluster module in the accelerator contains 2 AXI master interfaces and 1 AXI slave interface. The AXI transactions initiated by each of OpenRISC CPU cores inside the cluster, or by the central DMA units, will appear on either one of these two AXI master interfaces. If the destination address for the AXI transaction is in the address range of the accelerator (e.g. its destination is the *L2* memory or the other cluster module) then the AXI transaction will appear on the M_AXI_INT port, otherwise it will be routed to the M_AXI_EXT.

In fact, whenever one of the OpenRISC cores, or the DMA engines inside one of the cluster modules needs to fetch a chunk of data or program code from the shared DRAM memory of the system, its initiated transaction will be routed over the M_AXI_EXT port. These ports are connected to the HP0 slave port of the ZYNQ PS which can be used to access the shared DRAM in the system. The *RAB Table* unit in the figure is responsible for performing address translations on the incoming transactions from M_AXI_EXT before passing them to HP0 if needed.

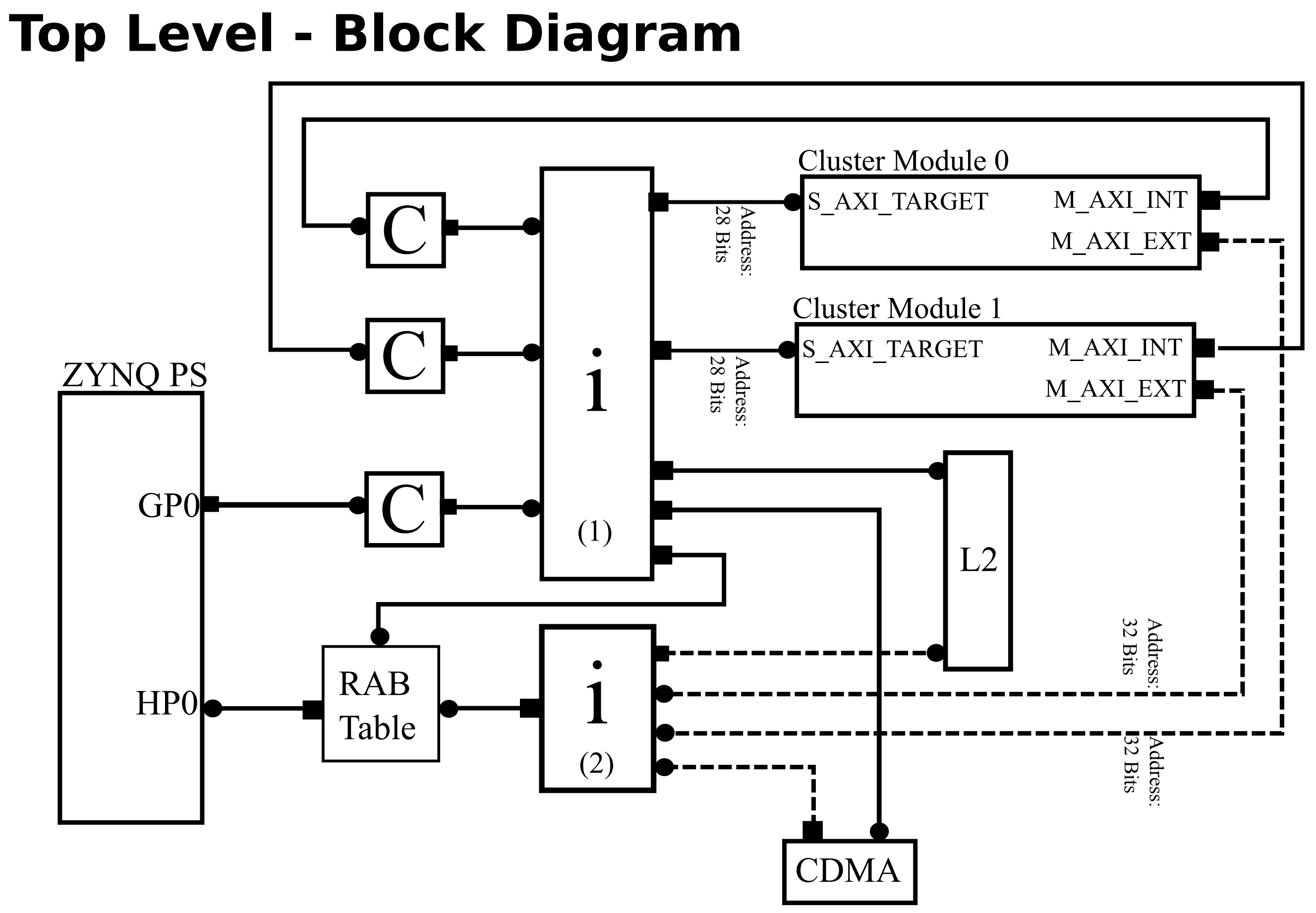


Figure 6.1: Developed architecture containing the ARM host (ZYNQ PS) connected to the L2 memory and two cluster modules. Each cluster module contains 4 OpenRISC cores. The RAB Table is responsible for translating addressese before passing them to the HP0 port of ZYNQ PS.

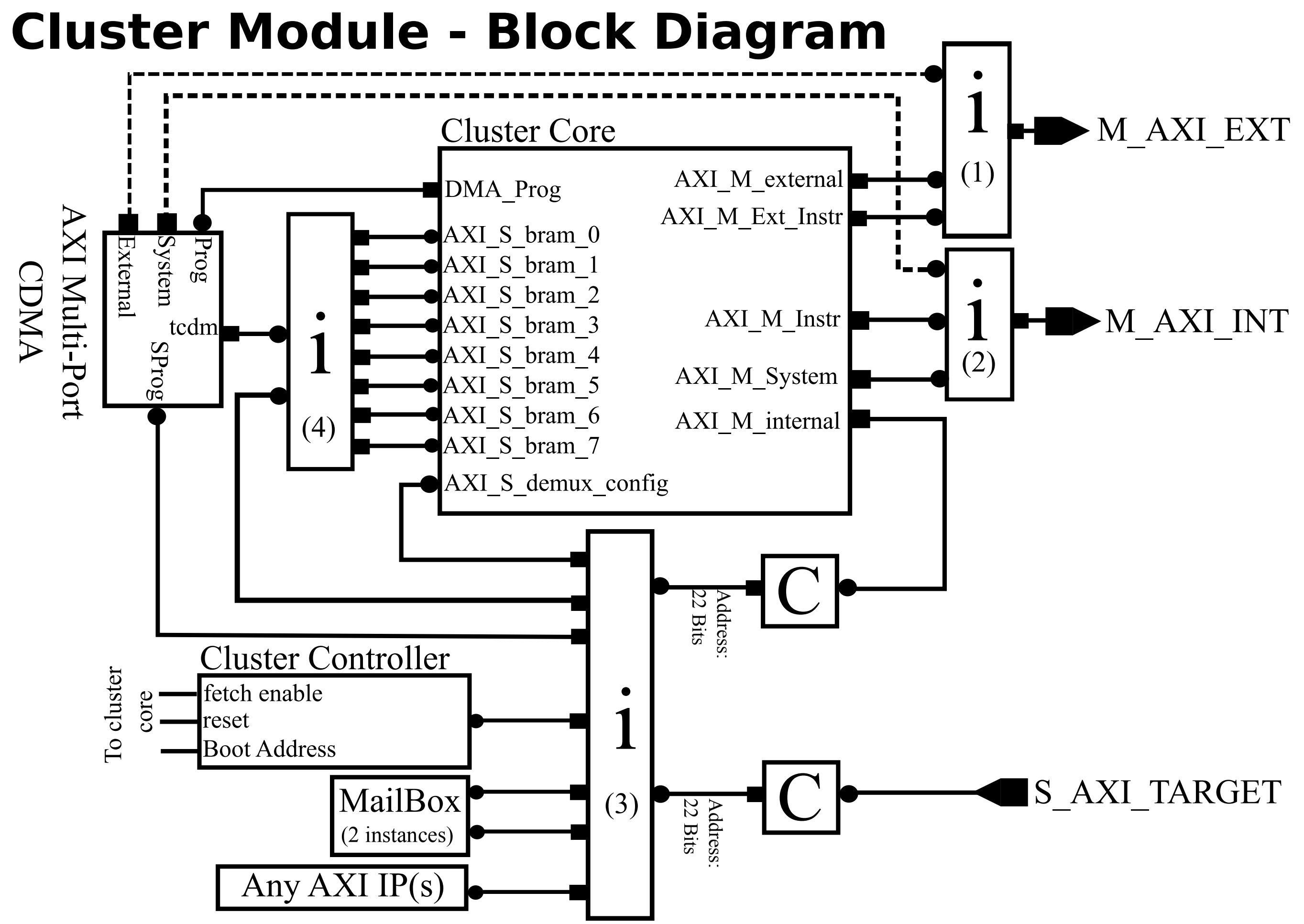


Figure 6.2: The implemented architecture for one OpenRISC cluster module. It contains a cluster core unit and all of the required AXI interconnects as well as local cluster peripherals. The AXI multi-port central DMA is and its connection to the second port of TCDM memories is also depicted.

In fact, the use of *RAB Table* enables us to isolate the address space used inside the system of OpenRISC clusters, from the physical address map used by the ARM host. It should be noted that the design of the cluster module is so that, the OpenRISC cores can either directly execute programs from the shared DRAM memory (which is also called *L3*) or the *L2* memory which is a part of the cluster system.

The GP0 port of the ZYNQ PS is utilized to perform direct access to the address space of the cluster system. Using GP0, the ARM host running on the ZYNQ PS is capable of programming the *L2* memory directly. As we show, using this port the ARM host can also have direct access to the TCDM memories inside each cluster, as well as all of the local peripherals of each of clusters. Futhermore, through GP0 the ARM host is capable of programming the DMA engines instantiated inside each cluster module to initiate data transfers from any point in the system to any other point.

Figure 6.2 represents the developed architecture for one *Cluster Module*. In the heart of the cluster module there exists the *Cluster Core* which contains the OpenRISC CPU cores. The *Cluster Core* unit constains a set of AXI interfaces:

- AXI_M_Instr: AXI master port through which the OpenRISC cores fetch program code from *L2* memory.
- AXI_M_Ext_Instr: AXI master port used by OpenRISC core to execute program code directly from *L3*.
- AXI_M_Internal: AXI master port used by OpenRISC cores to access local resources on the cluster mapped on the *alias* address range. We will further describe this in Section 6.2.1.
- AXI_M_System: AXI master port on which transactions in the address range of cluster system will be initiated.
- AXI_M_external: AXI master port dedicated to transactions to address ranges outside the address range of cluster system. (Such as *L3* memory.)
- AXI_S_demux_config: AXI slave port through which the ARM host can write key configuration parameters of each *Cluster Core* into its internal registers.

- AXI_S_bram: A group of AXI slave ports which allow the ARM host, or the OpenRISC cores in another cluster, or any DMA engine in the system to have direct access to TCDM memory blocks of the *Cluster Core*.

In addition to the above AXI interfaces, there exists a direct connection between the *Cluster Core* and the DMA engine. This interface is called DMA_Prog. This port provides the OpenRISC cores with an ultra-low latency interface to program the DMA engine and to initiate data transfer tasks.

Each cluster also contains a set a AXI peripherals connected to the local AXI interconnect of the cluster. From the view point of OpenRISC cores inside each cluster, each of these peripherals will always be seen at a fixed address regardless of which cluster the OpenRISC core resides in. The S_AXI_TARGET AXI slave interface of the cluster allows accesses to the local peripherals from outside world. Furthermore, through this port, incomming transactions can be routed to the AXI interconnect which is connected to the second port of TCDM memories.

As can be seen in Figure 6.2, all of the AXI transactions to the external address space are routed to the M_AXI_EXT port using an AXI interconnect. Similarly, all of the transactions targeted to the address space of the cluster system (which covers either another *Cluster Module* or the *L2* memory) will be routed to the M_AXI_INT port through another AXI interconnect.

The developed *AXI Multi-Port Direct Memory Access (DMA)* engine used in the cluster is a highly flexible and high performance data mover engine. Our developed DMA is capable of transfering data from any point in the entire system to any other point, with no exceptions. Our DMA engine can be programmed either through an ultra-low latency DMA_Prog port or its configuration AXI slave interface (called SProg). It contains an internal FIFO for the incoming commands. Assigning a task to the DMA engine is very simple: the source address, destination address and the length of data transfer should be written to the command FIFO of DMA engine one after another. As soon as the DMA engine is provided with a complete command it begins the transfer task and as soon as one transfer task is finished, it fetches and initiates the next transfer task if available.

The DMA engine has the following AXI interfaces:

- tcdm: An AXI master interface which is connected to the second port of TCDM memory blocks through an AXI interconnect.

- System: An AXI master interface used for transactions which are in the address range of the cluster system.
- External: An AXI master interface used for transaction which is outside the address range of the cluster system.
- SProg: Is an AXI slave port through which every master (either the ARM host, or an OpenRISC core in the system) can assign transfer tasks to the DMA engine.

Based on the given source and destination address values for a transfer task, the engine decides which of the AXI masters should be used for any of the source and destination transactions. The DMA engine contains an internal buffer of 8 KBytes. It first perforsm the read transaction and moves the data from the source address to its internal buffer, then it initiates a write transaction from the buffer to the destination port.

As shown in Figure 6.2, there exist AXI mailbox units which are the interfacing means between the host and the cluster when the host decides to offload a processing task to the cluster. One mailbox is dedicated to transferring the task pointers pushed by the host to the cluster. Another mailbox is used to pass the events and task execution results generated by cluster CPU cores to the host CPU. A separate AXI peripheral (which is called *Cluster Controller*) manages the reset and enable/disable status of each of the CPU cores in the cluster. It also provides each OpenRISC core with its own specific boot address. Thus the host has complete control over activity of each OpenRISC CPU core inside each of the clusters through this unit. Another unit is developed to monitor when each CPU core in the cluster finishes its assigned task and to update the host CPU with the situation through interrupts.

Figure 6.3 represents the architecture of the *Core Module*. The cluster core contains 4 OpenRISC 1K CPU cores. Each core contains a 64 Bits Wishbone3 [108] interface. We have developed a suitable bridge logic for translation of WishBone3 data transactions into equivalent AXI4 transactions. The translation allows OpenRISC CPU cores to talk to the developed AXI4 subsystem.

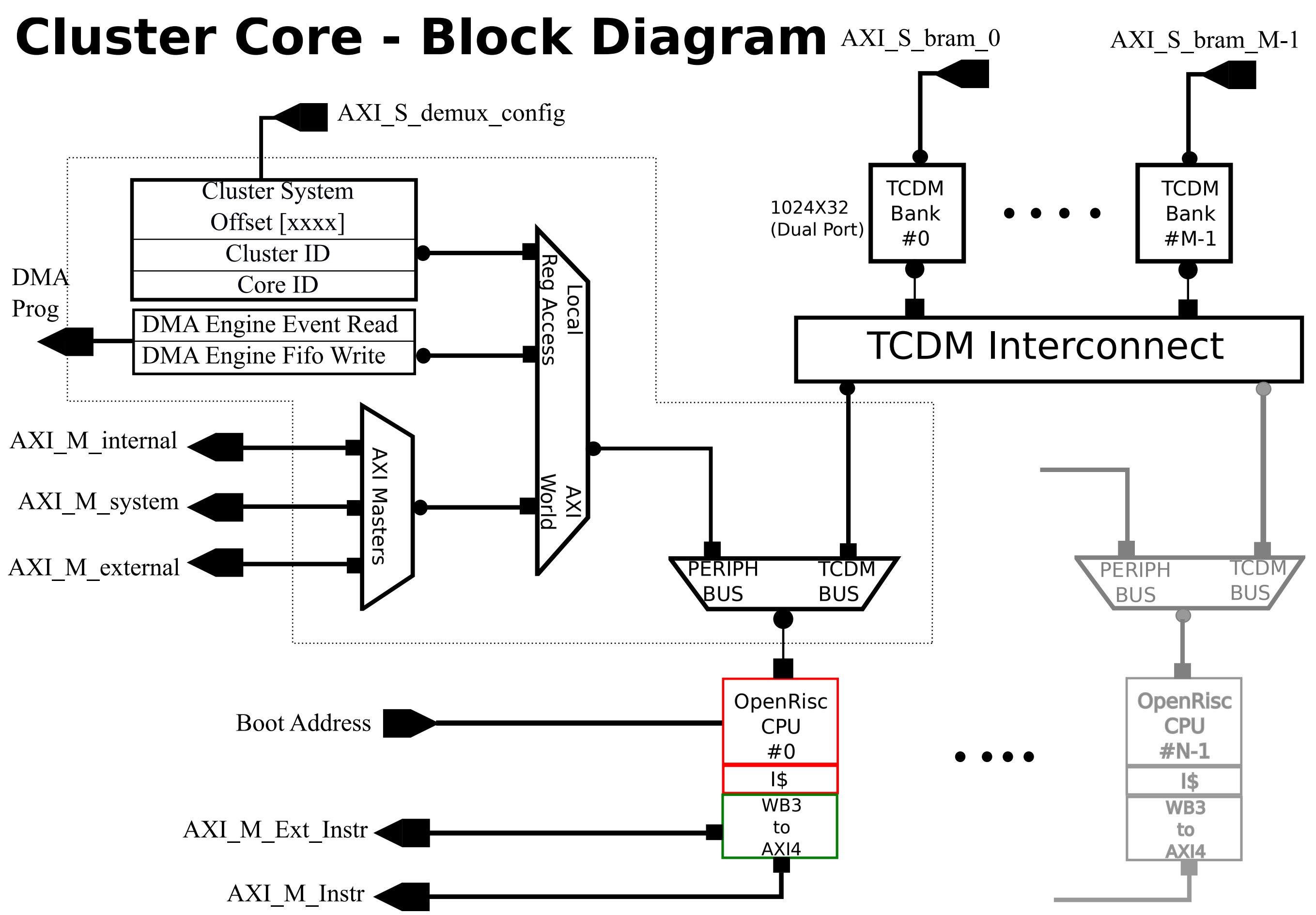


Figure 6.3: Architecture of one Cluster Core unit, which contains 4 instances of OpenRISC cores, 4 instances of demultiplexer unit, the logarithmic interconnect and tightly coupled memory blocks.

At data side of OpenRISC cores, the tightly coupled shared *L1* memory architecture is used. A logarithmic interconnect [93] is exploited to connect the CPU cores to the shared multi-banked data memory. In Figure 6.3 the logarithmic interconnect is depicted with *Log i*. The logarithmic interconnect allows the CPU cores to have single cycle read/write accesses to the shared data memory eliminating the need for a separate data cache. The number of memory banks residing on the logarithmic interconnect is a parametric value and currently is set to 8. There exists the possibility of attaching additional hardware accelerator units dedicated to specific computational tasks (such as FFT) to the logarithmic interconnect.

There exists a demultiplexer unit at the data port of each OpenRISC core. In Figure 6.3 the demultiplexer unit is indicated with a dashed line around it. This unit is responsible for monitoring the address of each of the generated transactions by the CPU core. It routes each transaction to its suitable destination according to the defined address map of the system. The demultiplexer units is also responsible for translating these transactions to suitable AXI transactions when the destination address is some where in the AXI subsystem. Furthermore, the demultiplexer of each OpenRISC core is responsible for providing it with the key configuration parameters of the core and the cluster. Examples of this key configuration values include the *core ID* of each core and the *cluster ID* to which each core belongs.

When the destination address is pointing to the shared *L1* memory, the transactions are passed through the TCDM interface of the demultiplexer to the logarithmic interconnect. In all of the other cases the transactions are routed to a second demultiplexer unit. When the OpenRISC needs to access a specific peripheral or resource located inside the same cluster, the AXI_M_internal plug will handle the transaction. However, if the OpenRISC requires access to resources of another cluster in the system or to the program memory, then the demultiplexer initiates an AXI transaction on the M_AXI_system port which is connected to AXI interconnects which eventually get connected to other cluster units as well as the cluster's program memory. When the OpenRISC needs to read or write to shared system DRAM, the demultiplexer initiates an AXI transaction on its M_AXI_External interface. As shown in Figure 6.1, this interface is connected to an AXI interconnect which is a master HP0 port.

Our architecture exploits the available potential of FPGA block memories: for both of the data memory banks and program memory the memory is instantiated as a full dual port memory and the second port is used to perform *direct memory accesses* in parallel with the operation of the CPU cores. Our developed *AXI Multi-Port Central DMA* unit is then instantiated in the system to handle the second port of these memory banks. As described, either of the host or the OpenRISC CPU cores can schedule DMA transfers to either of program or data memories of the cluster by programming these DMA engines. Futhermore, as stated, in addition to DMA units, the host has also direct access to both of program and data memory of the cluster through the GP0 AXI master port of the ZYNQ PS.

Each OpenRISC core has its own separate input port through which the host can define the boot address of the OpenRISC core. As shown in 6.2 the *WB3 to AXI4* unit which is resided on the instruction bus of each OpenRISC core, contains two AXI master ports. One of the ports is dedicated to fetching program code from *L2* memory located inside the cluster system. Another port allows the OpenRISC to directly execute programs stored on the shared system DRAM memory (L3).

6.1.2 Address Map and Dynamic Address Remapping

Our developed hardware architecture allows us to perform explorations on the efficient mechanisms of CPU-Accelerator address sharing. Conceptually, when the host decides to offload a processing task to an accelerator unit, it copies the data which should be processed by the accelerator on the shared DRAM memory, and then passes the physical address of the data array to the accelerator. The host CPU is using a memory management unit (MMU), consequently all of the processes running on the host, as well as all of the allocated data arrays are using virtual address values. Thus when the host decides to pass the address pointer of a data object to the accelerator, it first needs to convert the virtual address of that data object into its equivallent physical one.

In the cases where the shared data objects between the host CPU and accelerator are single continuous arrays of data, this address passing mechanism is efficient however, as soon as the complexity of the data objects which should be shared between the CPU and the accelerator increases, the process of virtual to physical address conversion gets more and more time consuming. As a result, we try to devise an architectural solution which allows the host CPU to pass the virtual address of the data objects directly to the hardware accelerator without the need to convert them into their equivallent physical address. The described architecture in Section 6.1.1 makes this operation possible through the use of suitable address conversion units.

As shown in 6.1 the AXI transactions directed to the shared DRAM memory pass the *RAB Table* unit. This unit can be programmed by the host CPU as well as the OpenRISC cores, and is responsible for translating the address of the incoming AXI transaction, which is basically a virtual address value, to its equivallent physical address before sending it out to the HP0 port. Indeed the address value which is passed to the cluster by the host throught the GP0 port or using the task allocation structures, are virtual values. At the same time the *RAB Table* gets programmed with suitable address translation entries, so that later, whenever during the operation, any of the cluster modules required to access the DRAM memory it gets redirected to a correct location on the DRAM.

This scenario however, may face a problem if the virtual address passed by the host CPU to the accelerator is within the physical address range of the local resources and peripherals of the accelerator itself. Indeed, we needed to devise a mechanism through which we can make sure that this problem does never happen.

Towards this, the hardware architecture that we have created has one special capability: the base address for the entire cluster system can be changed on-the-fly and during system operation, without the need for rebooting the ARM host CPU or reconfiguring the programmable logic. Indeed each time the host decides to use the accelerator for a specific computational task, it first puts the accelerator into reset mode, then it configures the base address for the entire accelerator system, then copies the code that OpenRISC cores should execute into the $L2$ or $L3$ memory and initializes the shared data objects and the mailbox contents inside the clusters. It then indicates the boot address of each of the OpenRISC cores in the accelerator and then it brings the accelerator out of reset state. From this point on, all of the OpenRISC cores inside the accelerator, and also the DMA engines will see and will use the newly defined address mapping.

Consequently, when an application running on the ARM host CPU needs to use the accelerator, it first allocates a memory in its virtual address space with the same size as the physical address range of the accelerator. It then uses the obetained virtual address to configure the accelerator and to move its physical base address value to the same value as the allocated virtual address. In our architecture since the physical address range for the accelerator is 256 MBytes, and address bus width is 32 bits, the upper four bits of the virtual address will be used as the upper four bits of the physical base address of the accelerator. From this point onward, whenever the application running on the ARM host allocates a memory to share a data object with the accelerator, it is completely sure that the virtual address of this new shared data object will not collide with the address range of the accelerator hardware itself.

These descriptions now clarify why in the proposed architecture in Section 6.1.1 we always use separate AXI ports for AXI transactions which remain internal to the cluster system and those AXI transactions which are targeted to go completely outside the address range of the accelerator. Moreover, addition of the *C* blocks at the enterance port of AXI interconnects as shown in Figures 6.1 and 6.2 is also inline with our goal. Indeed the *C* blocks always drop the upper bits of the address channels of an incoming AXI bus allowing the AXI interconnect to perform address decoding based on only lower bits of the address. This way we make sure that whenever one of the local resources of the accelerator is accessed the transaction reaches the correct destination regardless of the values of the upper bits of incoming address since in practice the upper bits are only indicating the base address for the entire accelerator hardware.

Figure 6.4 shows an example address map of the system. In this address map we have supposed that the ARM host has configured the upper 4 bits of address as $0x1$. Thus the physical address range of accelerator begins from $0x10000000$ and goes up to $0x20000000$. Each cluster occupies an address range of 4 MBytes. The *alias* address range will be used uniquely by all the CPU cores inside all clusters. Indeed, each OpenRISC core inside each cluster, can find its local cluster resources, such as TCDM memory, DMA engine and local cluster peripherals in a same address position as every other OpenRISC core. This is extremely important since, using this technique the OpenRISC core does not practically need to cacluate its access address whenever it wants to access one of the local resources of the cluster.

As an example, if an OpenRISC core wants to access the first memory location of TCDM, it can always find it at address $0x18000000$. However, if the same core wants to access the first location of TCDM memory of the other *Cluster Module*, then if the core belongs to *Cluster Module 0* it should perform an access to $0x10400000$, otherwise if the core belongs to *Cluster Module 1* it should initiate an access to $0x10000000$ to read the required memory location.

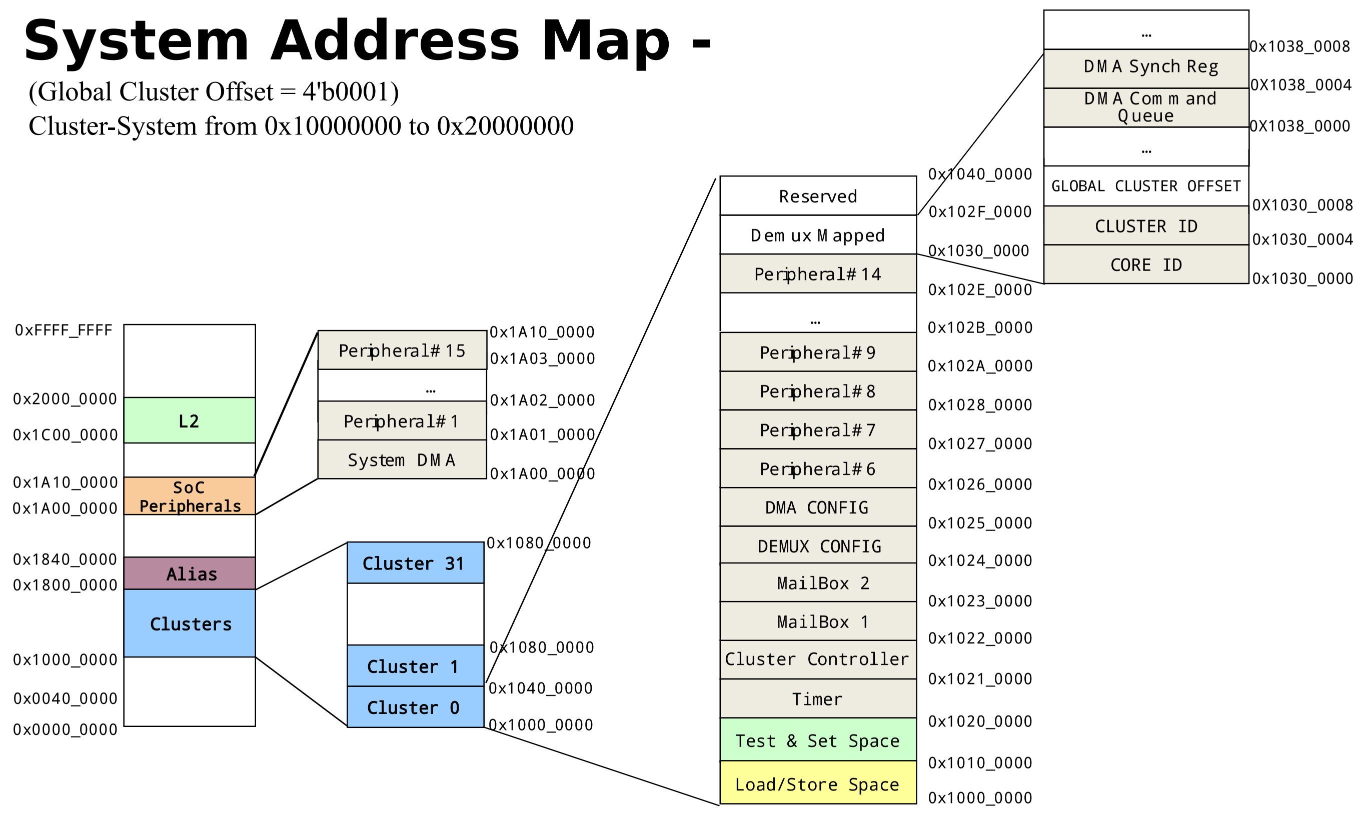


Figure 6.4: The address map of the system as seen by the components inside the accelerator.

As can be seen in Figure 6.4, the lowest address range for each cluster is occupied by its TCDM memories. A total TCDM memory range of 1 MBytes is considered for the design. Currenly, in the implemented hardware, total TCDM memory of each cluster is 512 KBytes. When an OpenRISC core needs to read its *Core ID* value, it just needs to perform a read access to the $0x18300000$ address in the memory. This access will be done in just one clock cycle since the these important registers are directly stored in the demultiplexer unit. It should be noted that the values of these important configuration registers can be only changed by the ARM host and not any unit inside the accelerator.

6.1.3 Practical Implementation

The hardware is realized on the Xilinx ZYNQ XC7Z045 device on the ZC-706 board. We have verified the correct functionality of this architecture in action using the real hardware and a set of benchmarks. Table 6.1 shows the resource utilization of the XC7Z045 device for our hardware architecture consisting two clusters. The running frequency for the hardware is 70 MHz.

Figure 6.5 shows the layout of the final routed design on the ZYNQ device. In this Figure, each of the major blocks in the hardware is highlighted with a different color. All of the points with the same color belong to a similar hardware block. Each block is identified using a marker.

Cluster Core unit which contains the OpenRISC cores and the TCDM, the interconnect logic inside each cluster and the DMA engines have the highest levels of resource usage in the design.

A Linux kernel level driver running on the ARM host is developed for interfacing with the cluster. This includes programing the direct memory access engines, transferring compiled OpenRISC executables to the shared program memory of the cluster, resetting and enabling cluster CPU cores and other similar tasks.

6.2 Future Work

As our ongoing research and the scheduled tasks to be done in future, we first adapt the developed infrastructure described in Section 6.1 to run OpenMP applications on

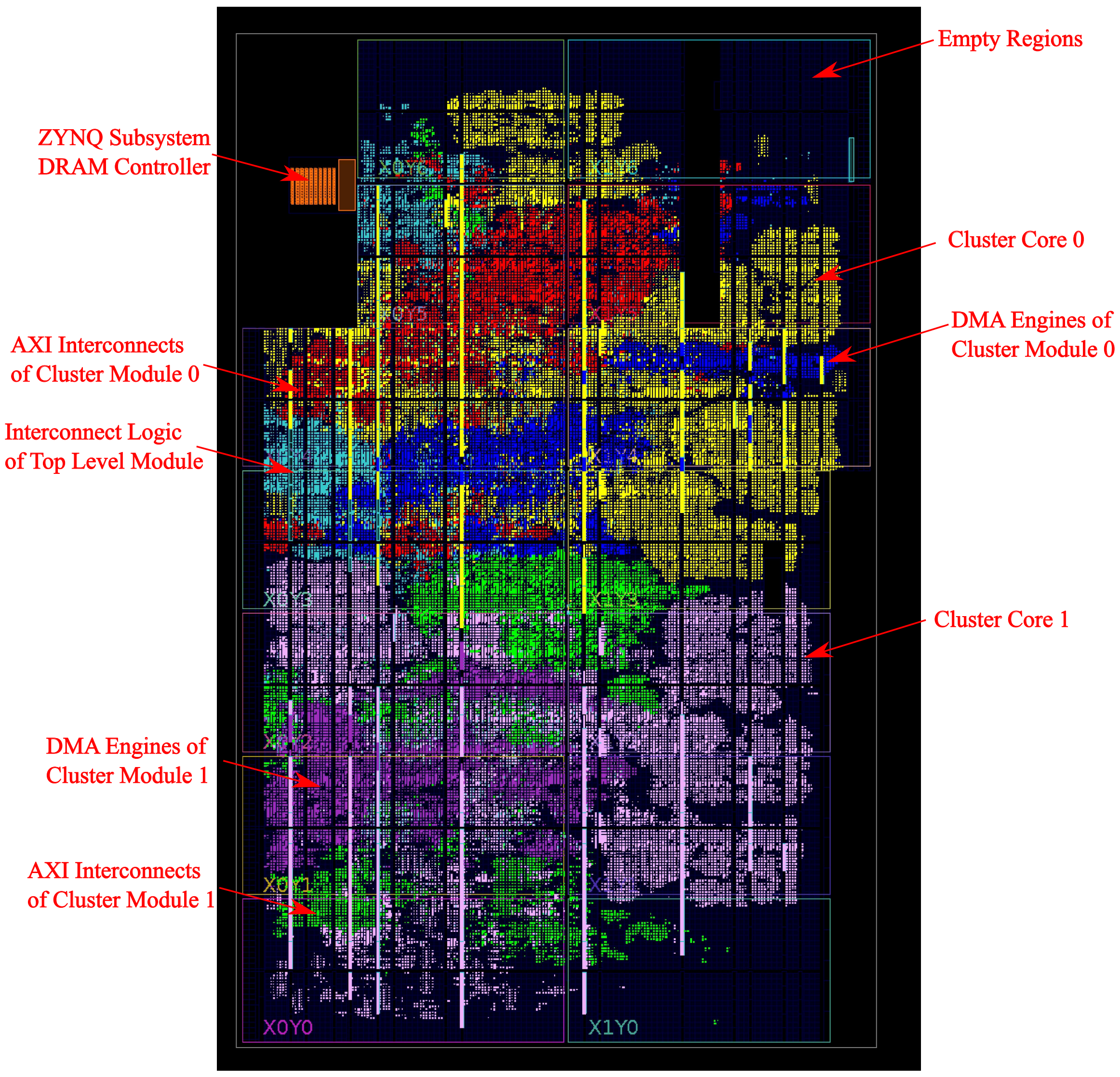

Empty Regions
ZYNQ Subsystem DRAM Controller
Cluster Core 0
DMA Engines of Cluster Module 0
AXI Interconnects of Cluster Module 0
Interconnect Logic of Top Level Module
Cluster Core 1
DMA Engines of Cluster Module 1
AXI Interconnects of Cluster Module 1
X0Y6
X0Y5
X0Y3
X1Y3
X1Y4
X0Y1
X0Y0
X1Y0

Figure 6.5: The layout of the final routed design on Xilinx XC7Z054 device. The building elements of each of the major blocks in the design are highlighted with different colors.

it. Then, we study the efficiency of various CPU-accelerator address sharing schemes using the created platform. Finally, we focus on addition of thermal sensors into the system and using the temperature readings while scheduling computational tasks to the hardware accelerator units.

Table 6.1: Resource utilization of XC7Z045 APSoC for a cluster containing 8 OpenRISC CPU cores and all required glue logic.

Resource	Utilization	Available	Utilization%
FF	97076	437200	22
LUT	136706	218600	63
Memory LUT	2141	70400	3
BRAM	300	545	55
DSP48	32	900	4
BUFG	11	32	34

6.2.1 Exploration of CPU-Accelerator Address Sharing Methods

We perform a detailed comparison between the traditional method of passing addresses to hardware accelerators and our own developed new method. Towards this, we prepare a set of computational benchmarks which operate on different types of data-sets. We then develope the required firmware for partially accelerating these computational tasks using our clusters of OpenRISC.

In the first test, we turn off the dynamic address remapping and address translation capability of our hardware architecture and each time that the host CPU wants to share a data chunk with the hardware accelerator, it first allocates the memory and calculates all of the required physical address values and pass them to the accelerator. We measure the averaged power consumption of the system for running each benchmark, as well as its processing bandwidth.

In the second test, we turn on the dynamic address remapping capability of our infrastructure. For each chunk of data that the host requires to share with the accelerator, it first performs memory allocation, and then performed required configuration in the *RAB Table* and passes directly the virtual address values to data arrays to the accelerator. We compare the measured performance and power in this case with previous case, for all of the benchmarks.

6.2.2 Implementation of Thermal Sensors

Implementation of thermal sensing circuits on a programmable logic is a well studied problem [94, 135]. The basic idea is to use a chain of Look Up Table (LUT) instances to create a ring oscillator circuit. The oscillating output of the circuit acts as the clock input for a counter. The rate of change in the counter value is sampled during fixed intervals. Based on the readings the temperature of that specific physical coordinates of the FPGA is estimated.

Another proposed solution is the thermal sensing method introduced by Xilinx for its soft DRAM controller cores [106]. The architecture is based on a tap delay circuit. Figure 6.6 shows the circuit. First, a continuous chain of inverters (Taps) is created using LUTs. The output of each tap is sampled by a flip-flop and compared with the output of the next tap. The comparison result is again captured by another flip-flop. The whole circuit is driven by a clock source with a fixed clock frequency. In fact, the clock signal which is the input to the chain of taps, is also driving the flip-flops.

Figure 6.7 shows example waveforms of the circuit. Depending on the delay of each tap, and the period of the input clock, a fixed number of LUTs are in a clock phase. For example, if the clock input frequency is 166 MHz and the LUT delay is 620 ps then 5 LUTs will be in one clock phase. Thus the output of the circuit will be a constant pattern. By the change in temperature the delay of LUTs will change however the clock period is fixed since it is driven by a clock oscillator from outside. This results the number of LUTs in one clock phase to change. As a result the output pattern of the circuit will change. This will be used to estimate the temperature.

We implement both of the aforementioned circuits and perform the calibration step for both of them. A similar procedure as disscussed in Section 2.4.3 will be used for this purpose. We then perform a set of evaluations and compare the performance and accuracy of thermal sensors. The thermal sensor with better performance will be selected as the online temperature sensing mechanism for our infrastructure. We then create a network of temperature sensors distributed across the entire fabric. This allows us to monitor on-die temperature with acceptable spatial accuracy.

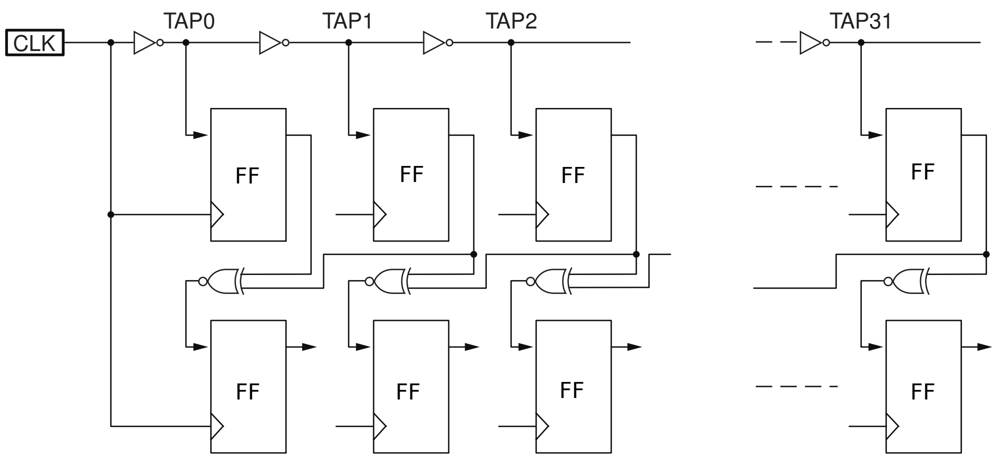


Figure 6.6: Tap delay circuit built using LUTs for temperature sensing.

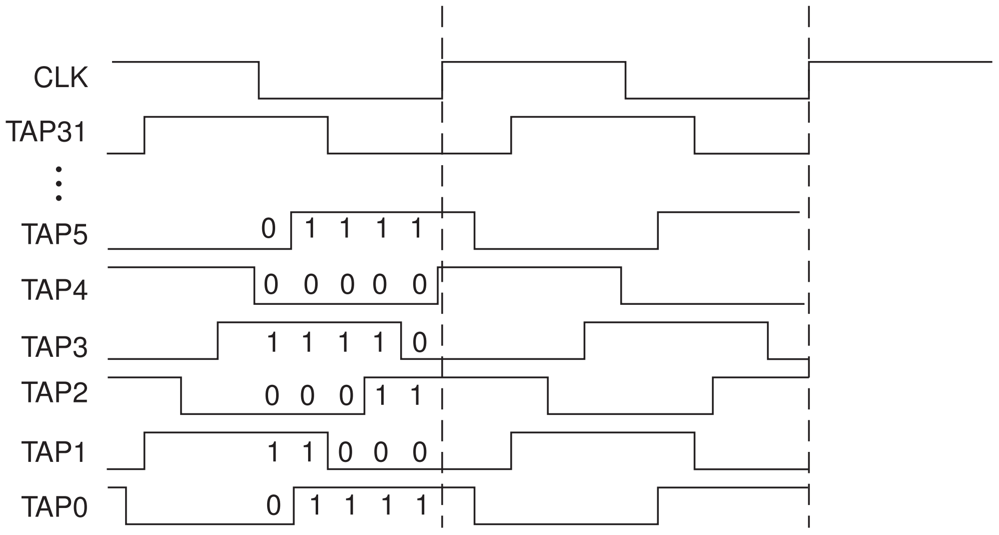


Figure 6.7: Sample waveforms of tap delay circuit operation.

6.2.3 Temperature Aware Assignment of Tasks to Accelerators

The idea of task migration between different CPU cores in a many-core architecture is not new [36, 48].

We study the problem for heterogeneous architectures which feature reconfigurable hardware. Basically, the host can decide weather to offload a computational task to the accelerator cluster or to run the task itself, based on its temperature readings.

Moreover, when assigning a task to computational units in a cluster, the host can decide the units which will be involved in the computation according to the device temperature map. Furthermore, there exist the possibility of reconfiguring some portions of the cluster according to the obtained temperature map and the task being executed.

Finally it should be noted that some recent FPGAs created using the FinFET technology (similar to Achronix [15]) are already running in temperature inversion region. This means that the hardware will run faster by increasing the temperature. This changes the decision making logic for task assignment completely. In fact, if a task is critical, and should be executed withing the smallest possible duration, it should be assigned to the CPU cores located in the hottest area of the fabric.

Chapter 7

Conclusions

We introduced the concepts of chip/package thermal modeling in Chapter 2. We described various available thermal models. We developed a thermal model for SCC chip using Hotspot thermal model. We then presented a calibration approach for the SCC on-die thermal sensors. We performed comparisons between the calibrated thermal sensor readings and estimations provided by thermal model. Comparisons show an error below 1.5°C.

We then analyzed the impact of frequency changes on the performance of many-core systems with MPI. We designed a large set of experiments using the monitoring infrastructure. Our analysis demonstrates that the communication patterns have a significant impact on performance and energy efficiency, varying from $0.1\times$ to $2\times$, compared to the baseline case without frequency perturbations.

In Chapter 3 we presented MiMAPT, a system capable of performing power, temperature and delay analysis of digital ICs while considering on-die temperature variation. The importance of considering temperature variation for delay/power analysis was demonstrated by practical examples. We showed that the sensitivity of delay and power estimations to temperature variation grows with scaling down the fabrication technology. We also quantified the estimation error that has to be expected when assuming an uniform temperature across the die. We highlighted several example cases in which this assumption leads to major inaccuracies. We also presented the multi-scale analysis capability of MiMAPT and its joint operation with a temperature variation aware power/delay estimation engine. Our results show that speed-ups can be obtained for thermal analysis while keeping the error of temperature estimation below

0.02 °C.

MiMAPT is implemented in Python and is available as a completely stand-alone and fully functional software. It provides an easy-to-use interface for designers to perform analysis without getting into details. Using the *merged virtual chip analysis* capability of MiMAPT, the developer can perform detailed analysis of the complete design before the final chip is implemented. The MiMAPT flow can be easily adapted to 3D-ICs. Considering the fact that performing thermal analysis for a 3D-IC with different spatial granularities for different silicon layers is not an easy to handle problem, the multi-granularity thermal floorplan creation feature of MiMAPT at gate-level may no more be used for 3D-ICs. In spite of this, as we showed in Section 3.5 MiMAPT is still faster compared to fine-grain method thanks to its early analysis at RTL.

During Chapter 4, We demonstrated the temperature variation aware bank-wise refresh which improves the overall refresh rate of the system effectively and decreases the power consumption of Wide-I/O DRAMs. We presented a thermal management scenario which results in improved energy consumption while keeping the temperature below the specified thresholds. To prove our ideas, we presented a virtual infrastructure featuring a TLM environment and detailed power and thermal models for the 3D chip. To speedup simulations, we devised a method to tune the sampling interval of simulation adaptively according to the thermal profile of the chip. We also studied the feasibility of hardware acceleration of the thermal simulation using the Maxeler engine.

In Chapter 6, we demonstrated an assessment on the performance of CPU-accelerator data sharing over AXI interfaces. We also presented our OpenRISC based cluster intended to be used for our future research on efficient CPU-accelerator address sharing as well as temperature variation aware hardware acceleration. We selected Xilinx *ZYNQ* APSoC as the target to develop required infrastructures.

Based on the obtained results in Chapter 6 (Section 5.3.5), we derive the following design rules:

- If a specific task should be done by cooperation of CPU and accelerator: The speed and energy consumption of *CPU ACP* and *CPU OCM* methods are always better than (or in the worst case equal to) *CPU HP*.
  The main drawback of using *CPU ACP* is that parts of the available cache space will be occupied by the acceleration task. Thus if there exists any other critical

application whose performance is heavily dependent on CPU caches, it may face problems handling its duty on-time. In this case, using *CPU OCM* (for small array sizes) and *CPU HP* (for big arrays) is recommended.

- If the task should be done by the hardware accelerator only (and then the CPU will just use the final result), then *ACP Only* or *ACP OCM* might be used only when the processed array blocks are small (smaller than the size of available cache, or on-chip memory) and there is no other background application consuming these resources. Otherwise, *HP Only* always provides better results.

The above rules can also be expanded to other platforms with similar architectures as the *ZYNQ*.

Finally, we introduced our multi-core heterogeneous architecture for the Xilinx ZYNQ device. Our architecture features a hardware accelerator which contains several clusters of OpenRISC CPUs. The special capability of the developed hardware architecture is that its address mapping can change dynamically and during system operation without the need to reboot the system or to re-generate the FPGA bitstream. This eventually allows the accelerator hardware to accept virtual address pointers to the shared data objects directly from the host CPU. We verified the correct functionality and operation of this platform on the Xilinx ZC-706 board. Our architecture provides us with the possibility of extending our research on efficient mechanisms of CPU-Accelerator data and address sharing as well as studying the ideas regarding temperature variation aware hardware acceleration.

Publications

[1] Andrea Bartolini, MohammadSadegh Sadri, Francesco Beneventi, Matteo Cacciari, Andrea Tilli, and Luca Benini. A system level approach to multi-core thermal sensors calibration. In JoséL. Ayala, Braulio García-Cámara, Manuel Prieto, Martino Ruggiero, and Gilles Sicard, editors, *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation*, volume 6951 of *Lecture Notes in Computer Science*, pages 22–31. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24153-6. doi: 10.1007/978-3-642-24154-3_3. URL http://dx.doi.org/10.1007/978-3-642-24154-3_3.

[2] Andrea Bartolini, MohammadSadegh Sadri, J. Furst, A.K. Coskun, and L. Benini. Quantifying the impact of frequency scaling on the energy efficiency of the single-chip cloud computer. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 181–186, 2012. doi: 10.1109/DATE.2012.6176459.

[3] Matthias Jung, Christian Weis, Mohammadsadegh Sadri, Norbert Wehn, and Luca Benini. SUBMITTED: optimized active and power-down mode refresh control in 3d-drams. In *Very Large Scale Integration (VLSI-SoC), 22th ACM/IEEE*, 2014.

[4] MohammadSadegh Sadri, Andrea Bartolini, and Luca Benini. Single-chip cloud computer thermal model. In *Thermal Investigations of ICs and Systems (THERMINIC), 17th International Workshop on*, pages 1 –6, sept. 2011.

[5] Mohammadsadegh Sadri, Andrea Bartolini, and Luca Benini. Mimapt: Adaptive multi-resolution thermal analysis at rt and gate level. In *Thermal Investigations of ICs and Systems (THERMINIC), 2012 18th International Workshop on*, pages 1–6, 2012.

[6] Mohammadsadegh Sadri, Christian Weis, Norbert Wehn, and Luca Benini. Energy and performance exploration of accelerator coherency port using xilinx zynq. In *Proceedings of the 10th FPGAworld Conference*, FPGAworld '13, pages 5:1–5:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2496-0. doi: 10.1145/2513683.2513688. URL http://doi.acm.org/10.1145/2513683.2513688.

[7] Mohammadsadegh Sadri, Andrea Bartolini, and Luca Benini. Temperature variation aware multi-scale delay, power and thermal analysis at rt and gate level. *Integration, the VLSI Journal*, 2014.

[8] Mohammadsadegh Sadri, Matthias Jung, Christian Weis, Norbert Wehn, and Luca Benini. Energy optimization in 3d mpsocs with wide-i/o dram using temperature variation aware bank-wise refresh. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2014*, 2014.

References

[9] Opencores benchmarks, . URL http://opencores.org.

[10] Reference removed for blind review. In *Proc. of RAPIDO 2013*, .

[11] Reference removed for blind review. In *Proc. of SNUG 2013*, .

[12] SmartBench: A multi-core friendly benchmark application, 2011. URL play.google.com/store/apps/details?id=com.smartbench.eleven.

[13] BLAS - Netlib, http://www.netlib.org/blas/, 2013. URL www.netlib.org/blas/.

[14] Reference removed for blind review. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2013.

[15] Achronix Semiconductor Corporation. Process technology, 2013. URL http://www.achronix.com/technology/process-technology.html.

[16] C. Albrecht. Iwls 2005 benchmarks. Technical report, IWLS, 2005. URL http://iwls.org/iwls2005/benchmark_presentation.pdf.

[17] N. Allec, Z. Hassan, Li Shang, R.P. Dick, and Ronggui Yang. Thermalscope: Multi-scale thermal analysis for nanometer-scale integrated circuits. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pages 603–610, 2008. doi: 10.1109/ICCAD.2008.4681639.

[18] L. Benini, E. Flamand, D. Fuin, and D. Melpignano. P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator.

In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 983–987, 2012. doi: 10.1109/DATE.2012.6176639.

[19] James Bennett, Ajay Bhasin, Jamila Grant, and Wen Chung Lim. Pid tuning via classical methods, Oct 2007. URL https://controls.engin.umich.edu/wiki/index.php/PIDTuningClassical.

[20] T.B. Berg. Maintaining i/o data coherence in embedded multicore systems. *Micro, IEEE*, 29(3):10–19, 2009. ISSN 0272-1732. doi: 10.1109/MM.2009.44.

[21] N. Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 Simulator. *SIGARCH Comput. Archit. News*, 39, 2011.

[22] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *High-Performance Computer Architecture (HPCA), The Seventh International Symposium on*, pages 171 –182, 2001. doi: 10.1109/HPCA.2001.903261.

[23] Cadence. *Low Power in Encounter RTL Compiler*. Cadence Design Systems, April 2010.

[24] Cadence. *Encounter Digital Implementation System User Guide*. Cadence Design Systems, March 2011.

[25] A. Calimera, R.I. Bahar, E. Macii, and M. Poncino. Temperature-insensitive dual-vth synthesis for nanometer cmos technologies under inverse temperature dependence. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(11):1608 –1620, nov. 2010. ISSN 1063-8210. doi: 10.1109/TVLSI.2009.2025884.

[26] C. Cascaval, Siddhartha Chatterjee, H. Franke, K.J. Gildea, and P. Pattnaik. A taxonomy of accelerator architectures and their programming models. *IBM Journal of Research and Development*, 54(5):5:1–5:10, 2010. ISSN 0018-8646. doi: 10.1147/JRD.2010.2059721.

[27] A. Chakraborty, K. Duraisami, P. Sithambaram, A. Macii, E. Macii, and M. Poncino. Thermal-aware clock tree design to increase timing reliability of embedded socs. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 57(10):2741 –2752, oct. 2010. ISSN 1549-8328. doi: 10.1109/TCSI.2010.2046959.

[28] Rajit Chandra, Adi Srinivasan, and Nanda Gopal. Thermally aware design modification, 10 2010. URL `http://www.patentlens.net/patentlens/patent/US_7823102/en/`.

[29] Rajit Chandra, Paolo Carnevali, John Yanjiang Shu, and Adi Srinivasan. Transient thermal analysis, 09 2011. URL `http://www.patentlens.net/patentlens/patent/US_8019580/en/`.

[30] K. Chandrasekar, C. Weis, B. Akesson, N. Wehn, and K. Goossens. System and Circuit Level Power Modeling of Energy-Efficient 3D-Stacked Wide I/O DRAMs. In *Proc. of DATE 2013*.

[31] ChipWorks Inc. A First Look at Apple A5 Processor, 2011. URL `http://www.chipworks.com`.

[32] Jongsok Choi, K. Nam, A. Canis, J. Anderson, S. Brown, and T. Czajkowski. Impact of cache architecture and interface on performance and area of fpga-based processor/parallel-accelerator systems. In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, pages 17–24, 2012. doi: 10.1109/FCCM.2012.13.

[33] Kihwan Choi, Ramakrishna Soma, and Massoud Pedram. Dynamic voltage and frequency scaling based on workload decomposition. In *ISLPED '04*, pages 174–179, 2004.

[34] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, and N. Wehn. Magali: A network-on-chip based multi-core system-on-chip for mimo 4g sdr. In *IC Design and Technology (ICICDT), 2010 IEEE International Conference on*, pages 74–77, 2010. doi: 10.1109/ICICDT.2010.5510291.

[35] R. Cochran and S. Reda. Consistent runtime thermal prediction and control through workload phase detection. In *Design Automation Conference (DAC), 47th ACM/IEEE*, pages 62 –67, june 2010.

[36] A.K. Coskun, T.S. Rosing, K.A. Whisnant, and K.C. Gross. Static and dynamic temperature-aware scheduling for multiprocessor socs. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(9):1127–1140, 2008. ISSN 1063-8210. doi: 10.1109/TVLSI.2008.2000726.

[37] G. Curren. Variability puts timing margins under pressure. *Electronics Systems and Software*, 2(1):30 – 33, feb.-march 2004. ISSN 1479-8336.

[38] J. W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A Supernodal Approach to Sparse Partial Pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20:720–755, 1999.

[39] G. Dhiman and T.S. Rosing. Dynamic voltage frequency scaling for multi-tasking systems using online learning. In *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, pages 207–212, 2007. doi: 10.1145/1283780.1283825.

[40] Gaurav Dhiman and Tajana Simunic Rosing. Dynamic voltage frequency scaling for multi-tasking systems using online learning. In *ISLPED '07*, pages 207–212, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-709-4. doi: http://doi.acm.org/10.1145/1283780.1283825. URL http://doi.acm.org/10.1145/1283780.1283825.

[41] Docea Power. Aceplorer: Esl solutions for low power design. URL http://www.doceapower.com/products-services/aceplorer.html.

[42] D. Dutoit et al. A 0.9 pj/bit, 12.8 gbyte/s wideio memory interface in a 3d-ic noc-based mpsoc. In *VLSI Technology (VLSIT), 2013 Symposium on*, pages C22–C23, 2013.

[43] EEMBC Inc. AndEBench: An EEMBC Benchmark for Android Devices, http://www.eembc.org/andebench/about.php, 2013. URL http://www.eembc.org/andebench/about.php.

[44] C.F. Fajardo, Zhen Fang, R. Iyer, G.F. Garcia, Seung Eun Lee, and Li Zhao. Buffer-integrated-cache: A cost-effective sram architecture for handheld and embedded platforms. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 966–971, 2011.

[45] Michael Ferdman et al. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. *SIGARCH Comput. Archit. News*, 40(1):37–48, March 2012. ISSN 0163-5964.

[46] Russell Fish. Future of computers: The power wall. *EDN Network*, Jan 2012. URL `http://www.edn.com/design/systems-design/4368858`.

[47] A. Fourmigue, G. Beltrame, G. Nicolescu, E.M. Aboulhamid, and I. O'Connor. Multi-granularity thermal evaluation of 3d mpsoc architectures. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1 –4, march 2011.

[48] Yang Ge, Qinru Qiu, and Qing Wu. A multi-agent framework for thermal aware task migration in many-core systems. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(10):1758–1771, 2012. ISSN 1063-8210. doi: 10.1109/TVLSI.2011.2162348.

[49] P. Ghosal, T. Samanta, H. Rahaman, and P. Dasgupta. Thermal-aware placement of standard cells and gate arrays: Studies and observations. In *Symposium on VLSI (ISVLSI), IEEE Computer Society Annual*, pages 369 –374, april 2008. doi: 10.1109/ISVLSI.2008.37.

[50] R. Gonzalez, B.M. Gordon, and M.A. Horowitz. Supply and Threshold Voltage Scaling for Low Power CMOS. *Solid-State Circuits, IEEE Journal of*, 32(8):1210–1216, 1997. ISSN 0018-9200. doi: 10.1109/4.604077.

[51] Google Inc. 0xBench: Comprehensive Benchmark Suite for Android, http://code.google.com/p/0xbench/, 2013. URL `http://code.google.com/p/0xbench/`.

[52] Ratnakar Goyal and Naresh Kumar. Current based delay models: A must for nanometer timing. In *Cadence CDNLive*, 2005.

[53] Peter Greenhalgh. big.little processing with arm cortex-a15 and cortex-a7. september 2011. URL http://www.arm.com/files/downloads/big_LITTLE_Final_Final.pdf.

[54] Junjun Gu, Gang Qu, and Lin Yuan. Enhancing dual-vt design with consideration of on-chip temperature variation. In *Computer Design (ICCD), IEEE International Conference on*, pages 542 –547, oct. 2010. doi: 10.1109/ICCD.2010.5647619.

[55] S. Gupta and S.S. Sapatnekar. Compact current source models for timing analysis under temperature and body bias variations. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(11):2104 –2117, nov. 2012. ISSN 1063-8210. doi: 10.1109/TVLSI.2011.2169686.

[56] S.P. Gurrum, D.R. Edwards, T. Marchand-Golder, J. Akiyama, S. Yokoya, J. Drouard, and F. Dahan. Generic Thermal Analysis for Phone and Tablet Systems. In *Proc. of ECTC 2012*.

[57] K. Haghdad and M. Anis. Power yield analysis under process and temperature variations. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(10):1794 –1803, oct. 2012. ISSN 1063-8210. doi: 10.1109/TVLSI.2011.2163535.

[58] Z. Hassan, N. Allec, Fan Yang, Li Shang, R.P. Dick, and Xuan Zeng. Full-spectrum spatial-temporal dynamic thermal analysis for nanometer-scale integrated circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(12):2276 –2289, dec. 2011. ISSN 1063-8210. doi: 10.1109/TVLSI.2010.2076351.

[59] Zyad Hassan. Thermal analysis for nanometer-scale integrated circuits. Master's thesis, University of Colorado, 2009.

[60] J. Howard, S. Dighe, S.R. Vangal, G. Ruhl, N. Borkar, S. Jain, et al. A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *Solid-State Circuits, IEEE Journal of*, 46(1):173 –183, jan. 2011. ISSN 0018-9200. doi: 10.1109/JSSC.2010.2079450.

[61] Wei Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan. Hotspot: a compact thermal modeling methodology for early-stage vlsi design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(5): 501 –513, may 2006. ISSN 1063-8210. doi: 10.1109/TVLSI.2006.876103.

[62] Wei Huang, M. Allen-Ware, J.B. Carter, E. Cheng, K. Skadron, et al. Temperature-aware architecture: Lessons and opportunities. *Micro, IEEE*, 31(3): 82 –86, may-june 2011. ISSN 0272-1732. doi: 10.1109/MM.2011.60.

[63] Ken ichi Shinkai, Masanori Hashimoto, and Takao Onoye. A gate-delay model focusing on current fluctuation over wide range of processvoltagetemperature variations. *Integration, the VLSI Journal*, 46(4):345 – 358, 2013.

[64] Altera Inc. Adding hardware accelerators to reduce power in embedded systems. september 2009. ISBN WP-01112-1.0. URL http://www.altera.com/literature/wp/wp-01112-hw-reduce-power.pdf.

[65] ARM Inc. Introducing neon development, 2009. URL http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dht0002a/BABCJFDG.html.

[66] ARM Inc. *ARM CoreLink MMU-400 System Memory Management Unit*, 2011. URL http://infocenter.arm.com/help/topic/com.arm.doc.ddi0472a/DDI0472A_corelink_mmu_400_r0p0_trm.pdf.

[67] ARM Inc. *Cortex-A9 MPCore Technical Reference Manual*, 2012. URL http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0460c/CIAIIJCE.html.

[68] ARM Inc. *AMBA AXI and ACE Protocol Specification*, February 2013. URL http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022e/index.html.

[69] Micron Technology Inc. TN-41-01: Calculating Memory System Power for DDR3. Technical report, 2007.

[70] Xilinx Inc. *LogiCORE IP ChipScope AXI Monitor (DS810)*, March 2011. URL http://www.xilinx.com/support/documentation/ip_documentation/chipscope_axi_monitor/v2_00_a/ds810_chipscope_axi_monitor.pdf.

[71] Xilinx Inc. *LogiCORE IP AXI Master Burst (DS844)*, June 2011. URL http://www.xilinx.com/support/documentation/ip_documentation/axi_master_burst/v1_00_a/ds844_axi_master_burst.pdf.

[72] Xilinx Inc. *ZC-702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC*, April 2013. URL http://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/ug850-zc702-eval-bd.pdf.

[73] Intel. *SCC External Architecture Specification (EAS)*, . URL http://communities.intel.com/community/marc. Revision 1.1.

[74] Intel. *Using the Sensor Registers*, . URL http://communities.intel.com/community/marc. Revision 1.1.

[75] Intel Inc. Intel Math Kernel Library, 2013. URL http://software.intel.com/en-us/intel-mkl.

[76] Canturk Isci, Gilberto Contreras, and Margaret Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *MICRO 39*, pages 359–370, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2732-9. doi: http://dx.doi.org/10.1109/MICRO.2006.30. URL http://dx.doi.org/10.1109/MICRO.2006.30.

[77] S. Ishikawa, A. Tanaka, and T. Miyazaki. Hardware accelerator for blast. In *Embedded Multicore Socs (MCSoC), 2012 IEEE 6th International Symposium on*, pages 16–22, 2012. doi: 10.1109/MCSoC.2012.22.

[78] S. Kaxiras and A. Ros. Efficient, snoopless, system-on-chip coherence. In *SOC Conference (SOCC), 2012 IEEE International*, pages 230–235, 2012. doi: 10.1109/SOCC.2012.6398353.

[79] A. Kennedy, Xiaojun Wang, and Bin Liu. Energy efficient packet classification hardware accelerator. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, 2008. doi: 10.1109/IPDPS.2008.4536216.

[80] Khronos Group. The open standard for parallel programming of heterogeneous systems: Opencl 2.0. URL http://www.khronos.org/opencl/.

[81] J.-S. Kim et al. A 1.2 V 12.8 GB/s 2 Gb Mobile Wide-I/O DRAM With 4*128 I/Os Using TSV Based Stacking. *IEEE Journal of Solid-State Circuits*, 47, 2012.

[82] J. Koppanalil, G. Yeung, D. O'Driscoll, S. Householder, and C. Hawkins. A 1.6 ghz dual-core arm cortex a9 implementation on a low power high-k metal gate 32nm process. In *Proc. of VLSI-DAT 2011*.

[83] George Kyriazis. Heterogeneous system architecture: A technical review. Technical report, Advanced Micro Devices, August 2012. URL http://developer.amd.com/wordpress/media/2012/10/hsa10.pdf.

[84] S. Lafond and J. Lilius. Interrupt costs in embedded system with short latency hardware accelerators. In *Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the*, pages 317–325, 2008. doi: 10.1109/ECBS.2008.39.

[85] Seri Lee. Optimum design and selection of heat sinks. In *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), Eleventh Annual IEEE*, pages 48 –54, feb. 1995. doi: 10.1109/STHERM.1995.512051.

[86] J. Levon, Maynard Johnson, et al. Oprofile: A system profiler for linux, 2013. URL http://oprofile.sourceforge.net.

[87] Peng Li, L.T. Pileggi, Mehdi Asheghi, and R. Chandra. Ic thermal simulation and modeling via efficient multigrid-based approaches. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(9):1763–1776, 2006. ISSN 0278-0070. doi: 10.1109/TCAD.2005.858276.

[88] Zuowei Li, Yuchun Ma, Qiang Zhou, Yici Cai, Yuan Xie, and Tingting Huang. Thermal-aware P/G TSV planning for IR drop reduction in 3D ICs. *Integration, the VLSI Journal*, 46(1):1 – 9, 2013.

[89] Weiping Liao, Lei He, and K.M. Lepak. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(7):1042 – 1053, july 2005. ISSN 0278-0070. doi: 10.1109/TCAD.2005.850860.

[90] J. Lin, Hongzhong Zheng, Zhichun Zhu, and Zhao Zhang. Thermal Modeling and Management of DRAM Systems. *IEEE Trans. on Computers.*, 2013.

[91] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu. RAIDR: Retention-Aware Intelligent DRAM Refresh. In *Proc. of ISCA 2012*.

[92] J. Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu. An Experimental Study of Data Retention Behavior in Modern DRAM devices: Implications for Retention Time Profiling Mechanisms. *SIGARCH Comput. Archit. News*, 41, 2013.

[93] Igor Loi. *Logarithmic Interconnect*, 2012.

[94] Sergio Lopez-Buedo and Eduardo Boemo. Making visible the thermal behaviour of embedded microprocessors on fpgas: A progress report. In *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, FPGA '04, pages 79–86, New York, NY, USA, 2004. ACM. ISBN 1-58113-829-6. doi: 10.1145/968280.968293. URL http://doi.acm.org/10.1145/968280.968293.

[95] George Mekhtarian. Composite current source (ccs) modeling technology backgrounder. Technical report, Synopsys, 2005.

[96] O. Mencer. Maximum performance computing for exascale applications. In *Proc. of SAMOS 2012*, 2012.

[97] J. Meng, K. Kawakami, and A.K. Coskun. Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints. In *Proc. of DAC 2012*.

[98] Mentor Graphics. *FloTHERM: Optimizing the Thermal Design of Electronics*, May 2011. URL http://www.mentor.com/products/mechanical/products/upload/flotherm.pdf.

[99] F.J. Mesa-Martinez, M. Brown, J. Nayfach-Battilana, and J. Renau. Measuring power and temperature from real processors. In *Parallel and Distributed Processing (IPDPS), IEEE International Symposium on*, pages 1 –5, april 2008. doi: 10.1109/IPDPS.2008.4536423.

[100] M. Meterelliyoz, J. P. Kulkarni, and K. Roy. Analysis of sram and edram cache memories under spatial temperature variations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(1):2 –13, jan. 2010. ISSN 0278-0070. doi: 10.1109/TCAD.2009.2035535.

[101] Micron Technology Inc. TN-10-08: Technical Note, LPDDR Thermal Implications for Die Stacks. Technical report, 2010.

[102] Micron Technology Inc. *4Gb: x16, x32 Mobile LPDDR3 SDRAM, http://www.micron.com/products/dram/mobile-lpdram*, July 2013. URL http://www.micron.com/products/dram/mobile-lpdram.

[103] Jayathi Y. Murthy, Sreekant V. J. Narumanchi, Jose' A. Pascual-Gutierrez, Tianjiao Wang, Chunjian Ni, and Sanjay R. Mathur. Review of multiscale simulation in submicron heat transfer. *International Journal for Multiscale Computational Engineering*, 3(1):5–32, 2005. ISSN 1543-1649.

[104] M. Nadeem, S. Wong, G. Kuzmanov, and A. Shabbir. A high-throughput, area-efficient hardware accelerator for adaptive deblocking filter in h.264/avc. In *Embedded Systems for Real-Time Multimedia, 2009. ESTIMedia 2009. IEEE/ACM/IFIP 7th Workshop on*, pages 18–27, 2009. doi: 10.1109/ESTMED.2009.5336814.

[105] S. Nazarian, H. Fatemi, and M. Pedram. Accurate timing and noise analysis of combinational and sequential logic cells using current source modeling. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(1):92 –103, jan. 2011. ISSN 1063-8210. doi: 10.1109/TVLSI.2009.2024945.

[106] Samson Ng. *DDR2 SDRAM Interface for Spartan-3 Generation FPGAs*, January 2009. URL http://www.xilinx.com/support/documentation/application_notes/xapp454.pdf.

[107] M. O'Connor. Accelerated processing and the fusion system architecture. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pages 93–93, 2012. doi: 10.1109/ASPDAC.2012.6165070.

[108] Opencores Group. Specification for the: Wishbone system-on-chip (soc) interconnection architecture for portable ip cores, September 2002. URL http://cdn.opencores.org/downloads/wbspec_b3.pdf.

[109] Opencores Group. *OpenRISC 1000 Architecture Manual*, December 2012. URL http://opencores.org/or1k/Architecture_Specification#Download_OR1K_1.0_Specification.

[110] OpenMP Architecture Review Board. Openmp application program interface version 4.0, July 2013. URL http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf.

[111] Suhas V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation, New York, 1980.

[112] X. Pu, Mikel Ash, Krishnaswamy Nagaraj, Joonsung Park, Steve Vu, Paul Kimelman, and Sean De La Haye. A +0.4 C accurate high-speed remote junction temperature sensor with digital Beta correction and series-resistance cancellation in 65nm CMOS. In *VLSI Circuits (VLSIC), 2013 Symposium on*, pages C214–C215, 2013.

[113] Sherief Reda, Abdullah N. Nowroz, Ryan Cochran, and Stefan Angelevski. Post-silicon power mapping techniques for integrated circuits. *Integration, the VLSI Journal*, 46(1):69 – 79, 2013.

[114] P. Rodgers and V. Eveloy. Design challenges for high-performance heat sinks used in microelectronic equipment: evolution and future requirements. In *Thermal*

and Mechanical Simulation and Experiments in Microelectronics and Microsystems (EuroSimE), Proceedings of the 5th International Conference on, pages 527 – 529, 2004. doi: 10.1109/ESIME.2004.1304087.

[115] A. Rodrigues, Elliot Cooper-Balis, Keren Bergman, Kurt Ferreira, David Bunde, and K. Scott Hemmert. Improvements to the structural simulation toolkit. In *Proc. of SIMUTOOLS 2012*.

[116] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. DRAMSim2: A Cycle Accurate Memory System Simulator. *Computer Architecture Letters*, 10, 2011.

[117] J.A. Rowlette and K.E. Goodson. Fully coupled nonequilibrium electron-phonon transport in nanometer-scale silicon fets. *Electron Devices, IEEE Transactions on*, 55(1):220 –232, jan. 2008. ISSN 0018-9383. doi: 10.1109/TED.2007.911043.

[118] K. Skadron, T. Abdelzaher, and M.R. Stan. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proc. of HPCA 2002*.

[119] A. Sridhar, A. Vincenzi, M. Ruggiero, Thomas Brunschwiler, and D. Atienza. 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In *Proc. of ICCAD 2010*.

[120] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza. 3d-ice: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling. In *Computer-Aided Design (ICCAD), IEEE/ACM International Conference on*, pages 463 –470, nov. 2010. doi: 10.1109/ICCAD.2010.5653749.

[121] Nicolas Chaussade Stephane Eric Sebastien Brochier. Managing the storage of data in coherent data stores, 09 2009. URL `http://www.google.com/patents/US20090216957`.

[122] Taeweon Suh, D.M. Blough, and H.-H.S. Lee. Supporting cache coherence in heterogeneous multiprocessor systems. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 1150–1155 Vol.2, 2004. doi: 10.1109/DATE.2004.1269047.

[123] Synopsys. Tluplus Q&A. Technical report, Synopsys. URL http://solvnet.synopsys.com.

[124] Synopsys. *PrimeTime Fundamentals User Guide*. Synopsys, December 2011.

[125] Synopsys. *Design Compiler User Guide*. Synopsys, December 2011.

[126] Taiwan Semiconductor Manufacturing Company. *TSMC N40LP Standard-cell Library Databook*, 2008. Version 120b.

[127] Taiwan Semiconductor Manufacturing Company. *TSMC 65nm Core Library Databook*, 2008. Version 200a.

[128] Zhanghong Tang. Hotspot detection by improved adaptive finite element method and its application in high-speed pcb and ic package design. *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, 2(10): 1659–1665, 2012. ISSN 2156-3950. doi: 10.1109/TCPMT.2012.2193146.

[129] Y. Temiz, M. Zervas, C. Guiducci, and Y. Leblebici. A CMOS-compatible chip-to-chip 3D integration platform. In *Proc. of ECTC 2012*, 2012.

[130] Andras Timar and Marta Rencz. Temperature dependent timing in standard cell designs. In *Thermal Investigations of ICs and Systems (THERMINIC), 18th International Workshop on*, sep. 2012.

[131] A. Timar, G. Bognar, A. Poppe, and M. Rencz. Electro-thermal co-simulation of ics with runtime back-annotation capability. In *Thermal Investigations of ICs and Systems (THERMINIC), 16th International Workshop on*, pages 1 –6, oct. 2010.

[132] TLMCentral Group. Tlm central. URL http://www.tlmcentral.com/.

[133] C.K. Turnes and J. Romberg. Spiral fft: An efficient method for 3-d ffts on spiral mri contours. In *Image Processing (ICIP), 17th IEEE International Conference on*, pages 617 –620, sept. 2010. doi: 10.1109/ICIP.2010.5653254.

[134] Rob F. van der Wijngaart, Timothy G. Mattson, and Werner Haas. Light-weight communications on intel's single-chip cloud computer processor. *SIGOPS Oper. Syst. Rev.*, 45:73–83, February 2011. ISSN 0163-5980. doi: http://doi.acm.org/

10.1145/1945023.1945033. URL http://doi.acm.org/10.1145/1945023.1945033.

[135] S. Velusamy, Wei Huang, J. Lach, M. Stan, and K. Skadron. Monitoring temperature in fpga based socs. In *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, pages 634–637, 2005. doi: 10.1109/ICCD.2005.78.

[136] R.K. Venkatesan, S. Herr, and E. Rotenberg. Retention-aware placement in DRAM (RAPID): software methods for quasi-non-volatile DRAM. In *Proc. of HPCA 2006*.

[137] Po-Hsun Wu and Tsung-Yi Ho. Bus-driven floorplanning with thermal consideration. *Integration, the VLSI Journal*, 46(4):369 – 381, 2013.

[138] Jianyong Xie, Daehyun Chung, M. Swaminathan, M. Mcallister, A. Deutsch, Lijun Jiang, et al. Electrical-thermal co-analysis for power delivery networks in 3d system integration. In *3D System Integration, 2009. 3DIC 2009. IEEE International Conference on*, pages 1 –4, sept. 2009. doi: 10.1109/3DIC.2009.5306525.

[139] Xilinx Inc. *Xilinx Power Estimator User Guide (UG440) http://www.xilinx.com/cgi-bin/docs/rdoc?v=latest;d=ug440-xilinx-power-estimator.pdf*, June 2013. URL http://www.xilinx.com/cgi-bin/docs/rdoc?v=latest;d=ug440-xilinx-power-estimator.pdf.

[140] Xilinx Inc. *Zynq-7000 All Programmable SoC Technical Reference Manual (UG585)*, March 2013. URL http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[141] Y. Yang, Z. Gu, C. Zhu, R. P. Dick, and L. Shang. Isac: Integrated space-and-time-adaptive chip-package thermal analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(1):86 –99, jan. 2007. ISSN 0278-0070. doi: 10.1109/TCAD.2006.882589.

[142] Kuo ying Tsai et al. Thermal characterization of a wide i/o 3dic. In *Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT), 2011 6th International*, pages 261–264, 2011.