

**MICROARCHITECTURAL LEVEL POWER ANALYSIS AND
OPTIMIZATION IN SINGLE CHIP PARALLEL COMPUTERS**

by

Priyadarshini Ramachandran

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in a partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

James M. Baker, Chairman

Dong S. Ha

Joseph G. Tront

July 2004

Blacksburg, Virginia

Keywords: Power Estimation, Architectural Level Analysis, Single Chip
Parallel Computers, Switching Capacitance, Leakage Current, SCMP

Copyright 2004, Priyadarshini Ramachandran

MICROARCHITECTURAL LEVEL POWER ANALYSIS AND OPTIMIZATION IN SINGLE CHIP PARALLEL COMPUTERS

Priyadarshini Ramachandran

James M. Baker, PhD, Committee Chair

Department of Electrical and Computer Engineering

Abstract

As device technologies migrate into Deep Submicron (DSM) feature sizes, high-performance power-efficient computer architectures that keep pace with improving technologies need to be explored. Technology scaling increases the effects of wire latencies, inductive effects, noise and crosstalk in on-chip communication, limiting the performance of DSM designs. Power efficient performance gains from Instruction Level Parallelism (ILP) are reaching a limit. Single-Chip Parallel Computers are promising solutions to the DSM design challenges and the performance limitations of ILP. These systems are explicitly modular architectures that efficiently support Thread Level Parallelism (TLP) while avoiding global signals and shared resources.

Microarchitectural level power analysis is required for evaluating the feasibility of newly conceived architectures in terms of power dissipation and energy efficiency. Accounting for power in the early stages of design shortens the time-to-market due to reduced design iteration times. Power optimizations at the architectural level can yield large power savings. This thesis proposes a microarchitectural level power estimation and analysis infrastructure for Single Chip Parallel Computers. The power estimation tool and the analysis methodology are developed based on the Single Chip Message-Passing Parallel (SCMP) Computer and can be extended to other Single Chip Parallel Computers. The thesis focuses on the development of power estimation models, construction of the power analysis tool, study of the power advantages of the architecture and identification of subsystems requiring power optimization.

This work was generously supported by the National Science Foundation (grant#: CCR-0113948)

Acknowledgements

I wish to offer my deepest sense of gratitude and appreciation to my advisor Dr. James M. Baker for his invaluable guidance and support throughout my graduate program. I would like to thank Dr. Dong S. Ha and Dr. Joseph G. Tront for serving on my committee. I also wish to thank all the members of the SCMP design team and my friends for the wonderful experience at Virginia Tech. I am deeply indebted to my parents, Andal and Manickam Ramachandran, for offering me enormous support, inspiration and encouragement to pursue my Master's degree. I would like to thank my fiancé, Balaji, for his constant encouragement and emotional support. I cannot forget to thank my brother, Karthik, for always believing in me and appreciating my work.

Contents

LIST OF CONTENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	ix
1. Introduction	1
1.1. Architectural Level Analysis and Optimizations	1
1.2. Design Challenges in Deep Submicron Technologies	2
1.2.1. Wire Delays	3
1.2.2. Inductance Noise and Delay Deterioration	4
1.2.3. Clock Distribution	4
1.3. Single Chip Parallel Computers	5
1.4. Related Work	6
1.5. Thesis Organization	7
2. Background	8
2.1. Power Dissipation in CMOS circuits	8
2.1.1. Capacitive Switching Power Dissipation	9
2.1.2. Short Circuit Power Dissipation	11
2.1.3. Leakage Power Dissipation	12
2.1.4. Static Power Dissipation	14
2.2. Architectural Level Power Estimation	14
2.2.1. Estimation of Dynamic Power Dissipation	14
2.2.2. Estimation of Static Power Dissipation	15
2.3. SCMP Architecture	16
2.3.1. CPU Architecture	17
2.3.2. Memory Subsystem	18
2.3.3. Network Architecture	18
2.3.4. Hardware Implementation	19

3. Power Models for SCMP	20
3.1. Dynamic Power Models for the CPU	20
3.1.1. Dynamic Power Model for Register Contexts	21
3.1.2. Dynamic Power Model for CMT & CMT Control	24
3.1.3. Dynamic Power Models for other components	27
3.2. Dynamic Power Models for the Network	28
3.2.1. Power Model for the Flit Buffer	28
3.2.2. Power Model for the Crossbar	29
3.2.3. Power Model for the Arbiter	30
3.3. Scaling Models to DSM Technologies	32
3.4. Leakage Power Modeling	34
3.4.1. Leakage Modeling for a Single Transistor	34
3.4.2. Leakage Modeling for a Design Cell	35
4. The Power Analysis Tool	37
4.1. The Performance Simulator	37
4.1.1. Processor Core Simulation	37
4.1.2. Network Simulation	39
4.1.3. Graphical Interface	40
4.2. The Power and Performance Simulator	42
4.2.1. Construction of the Power and Performance	42
4.2.2. Power and Performance Simulation	43
4.3. Power Analysis Tool Options	44
5. Power Analysis & Optimizations	49
5.1. Power Advantages in SCMP architecture	49
5.1.1. Thread Level Parallelism	49
5.1.2. Absence of Global Signals	50
5.1.3. Uniform Power Density	50
5.1.4. Power Performance Scaling	51

5.2. Power Optimizations	51
5.2.1. Split Register File	52
5.2.2. Clock Gating	52
5.2.3. Gated VDD	53
5.3. Power, Performance and Energy Efficiency	53
5.3.1. Power and Performance	54
5.3.2. Energy Efficiency	61
5.3.3. Design Trade-offs	64
6. Conclusions	67
6.1. Thesis Summary	67
6.2. Future Work	68
BIBLIOGRAPHY	69

LIST OF FIGURES

CHAPTER 2

Figure 2.1 CMOS Inverter	10
Figure 2.2 Equivalent Circuit for Charging & Discharging Capacitor	11
Figure 2.3 Sources of Leakage in a MOSFET	13
Figure 2.4 SCMP System with 64 tiles	17
Figure 2.5 Block Diagram of a SCMP Tile	18

CHAPTER 3

Figure 3.1 Wordlines and Bitlines in a SRAM Array Structure	22
Figure 3.2 A ' n to k ' Decoder	26
Figure 3.3 Wormhole Router	29
Figure 3.4 A 5×5 Matrix Crossbar Switch	31
Figure 3.5 Matrix Arbiter Modified for Round Robin Operation	32

CHAPTER 4

Figure 4.1 SCMP GUI Main View	40
Figure 4.2 SCMP GUI Node Detail View	41
Figure 4.3 Average Power Dissipation for Three Benchmarks	45
Figure 4.4 Peak Power Dissipated for the 64-node Configuration at 1 GHz	46
Figure 4.5 Distribution of Power Among Subsystems for a Single Node	47
Figure 4.6 Example of the Power Metering File	48
Figure 4.7 Graphical Representation of the Power Metering Data	48

CHAPTER 5

Figure 5.1a Performance for Transitive Closure at 5GHz	55
Figure 5.1b Power for Transitive Closure at 5GHz	55
Figure 5.1c Performance for Transitive Closure at 200MHz	56

Figure 5.1d Power for Transitive Closure at 200MHz	56
Figure 5.2a Performance for Neighborhood at 5GHz	57
Figure 5.2b Power for Neighborhood at 5GHz	57
Figure 5.2c Performance for Neighborhood at 200MHz	58
Figure 5.2d Power for Neighborhood at 200MHz	58
Figure 5.3a Performance for Conjugate Gradient at 5GHz	59
Figure 5.3b Power for Conjugate Gradient at 5GHz	59
Figure 5.3c Performance for Conjugate Gradient at 200MHz	60
Figure 5.3d Power for Conjugate Gradient at 200MHz	60
Figure 5.4 Energy Efficiency for Transitive Closure	62
Figure 5.5 Energy Efficiency for Neighborhood	63
Figure 5.6 Energy Efficiency for Conjugate Gradient	63
Figure 5.7 Energy Efficiency Vs Frequency	64
Figure 5.8 Distribution of Power Among Subsystems in SCMP	66

LIST OF TABLES

CHAPTER 3

Table 3.1 Wordline and Bitline Switching Capacitances	22
Table 3.2 Wire Dimensions Scaling	33
Table 3.3 Resistance Scaling	33
Table 3.4 Capacitance Scaling	33

CHAPTER 5

Table 5.1 Maximum Allowable Power, ITRS Roadmap	66
---	----

Chapter 1

1. Introduction

Single Chip Parallel Computers are emerging as a promising solution to the issues arising due to technology scaling and limits of Instruction Level Parallelism. Extensive design space exploration of such architectures requires architectural level evaluation of both power and performance. This thesis focuses on microarchitectural level power analysis and optimizations in Single Chip Parallel Computers. The SCMP (Single Chip Message-Passing Parallel) system is a single chip parallel computer designed for high-performance and power efficiency in Deep Submicron (DSM) technologies. The power analysis and optimization approach presented in this work is developed using the SCMP architecture and can be extended to other Single Chip Parallel Computers.

1.1. Architectural Level Analysis and Optimizations

In order to explore new computer architectures, designers need to analyze and evaluate processor architectures before building hardware prototypes. Simulator tools or software models of the proposed architectures, when used to execute realistic workloads, accelerate the design process by enabling easy and fast design space exploration. In the past architectural analysis tools were expected to analyze only the performance of the design. Recently, power dissipation has also become an important design criterion. Long a concern in mobile processors where battery life is limited, power dissipation is also important in high-performance processors where, energy supply is not typically a limitation, but increasing design complexity and operating speeds are causing power dissipation to reach the thresholds of current packaging technologies [1,2]. Increases in power dissipation also affect the reliability of integrated circuits [3]. High-level analysis enables thorough exploration of architectural level trade-offs that often result in large

delay and power savings [4]. Hence, computer architects need tools that can be used to perform architectural level power and performance analysis.

When typical power optimization opportunities and design iteration times required to perform power analysis at different levels of abstraction are studied, it is observed that power optimization opportunities are larger at higher levels with shorter design iteration times [5,6]. Architectural power management and optimizations can yield large power savings. With increasing levels of integration and increasing clock frequencies, power optimizations at the logic-level and transistor level are not sufficient to meet the power constraints of design. High-level power estimation is essential for analysis of architectural level design trade-offs. It is also required for identifying the power advantages in any new architecture.

In absence of high-level analysis tools the design must be synthesized to a gate level or transistor level net-list, which must be validated, and then a logic level or transistor level power analysis tool must be used to obtain the power consumption statistics [4]. The large time required to obtain the lower-level net list and the large run times of lower level analysis tools render such methods inefficient for exploring high-level design trade-offs. Using high-level analysis tools for exploring high-level design trade-offs leads to fewer and faster design iterations due to reduced complexity.

The reduced complexity of high-level analysis tools comes at the cost of lower absolute accuracy. However, in order to analyze high-level design trade-offs and optimizations relative accuracy is more important than absolute accuracy [7,8]. Hence, lower level tools can be limited to lower level optimizations and accurate power budget verifications, and high-level tools can be used for architectural level and system level power optimizations. By following this design strategy, a highly optimized design can be obtained with a reduced design cycle time.

1.2. Design Challenges in DSM Technologies

Over the years there has been a significant increase in integration density and computational complexity. This has been possible due to advances in device

manufacturing technologies that allow a steady decrease in transistor feature sizes. With decreasing feature sizes, switching speeds of transistors increase thereby enabling higher operating frequencies. Continued reductions in transistor feature sizes affect the operating characteristics and properties of the transistor. DSM geometries pose a number of problems that show the potential to stop the continuation of Moore's law which projects that the number of transistors that can be integrated on a chip doubles every 18 to 24 months. The major DSM impact is on on-chip interconnect. Significant changes in design methodologies are needed in order to face the challenges imposed by DSM technologies [9, 10]. Designers need to explore computer architectures that scale in power and performance to keep pace with the improving technologies.

1.2.1. Wire Delays

One of the most serious DSM problems is the rising RC on-chip interconnect wire delays. It has been studied that interconnect delays will be a dominant factor in the delay and performance budget of DSM designs [11, 12]. As feature sizes of transistors are reduced, the diameter of the interconnect wires have to shrink to match transistor size. With improving device technologies, there is an ever-increasing demand for higher integration densities. This fuels the need to drop wire pitches to match device dimensions. Wire resistance, which is inversely proportional to the cross-sectional area of the wire, increases with decreasing wire pitch, thereby increasing the RC wire delay [11]. These wire latencies will have an increasing effect on the global signals and interconnect delays of processors, limiting their performance.

It is observed that wires spanning relatively shorter distances, like a fixed number of gates, get shorter with technology scaling. Hence, their performance is not affected by scaling [12]. However, the lengths of global wires do not scale with technology scaling. Hence, on-chip global communications will involve long latencies. The RC delay of an un-interrupted wire increases quadratically with length. One solution to counter the large delays of the long global wires is to break the wires into a number of segments by introducing repeaters. But the use of repeaters has some setbacks [11,12]. Designs using repeaters are complicated. It is critical to place repeaters at optimal positions, which is often not feasible. In DSM designs large delays of chip-length wires require a number of

repeaters, which implies a large silicon area. Also, repeaters significantly increase the power dissipation of the chip [11].

1.2.2. Inductance, Noise and Delay Deterioration

The bandwidth of a digital signal depends on its rise time. In DSM designs, on-chip signal rise times decrease thereby increasing signal bandwidth. Hence, as technology scales, inductive effects will become prominent [11]. Overdriving of global lines can result in inductive problems like ringing effects and additional delays. Proper device sizing in large drivers and shielded wires are needed to eliminate problems due to inductive effects. Inductive parasitics and dominance of wire coupling capacitances cause noise and crosstalk to become significant issues in DSM designs. Rise in coupling capacitance causes delay deterioration because the effective load capacitance is no longer constant. Delay varies with neighboring signal activity making static timing analysis a challenge. Hence, it will be difficult to implement, analyze and optimize extensive and complicated designs using DSM technologies. Modular design styles in which the system is constructed from simple primitive modules will be easier to implement in deep submicron technologies.

1.2.3. Clock Distribution

With increase in die sizes and clock frequencies, as predicted [13], signal Time-Of-Flight (TOF) will limit processor speed and performance. For example, a system implemented in a 750mm^2 die, using the 50nm technology node cannot support a global clock frequency greater than 3.58GHz [11]. RC delays, inductance and reliable clock distribution are also factors limiting the maximum clock speed for a design. When clock period is decreased the allowable clock skew also decreases. While local clock skews will be manageable, global clock skew will turn out to be the bottleneck in the design. Providing a low skew, high-speed clock distribution across the die will be impossible in DSM technologies. New design methodologies that do not require low skew clock distribution across the entire area of the die must be looked at.

1.3. Single Chip Parallel Computers

As device technologies migrate into DSM feature sizes, computer architectures need to scale in power and performance to keep pace with the improving technology. Over the years, increases in processor performance have been achieved by increasing the clock frequency of the processor and by exploiting Instruction-Level Parallelism (ILP) to execute more instructions each clock cycle. A high level of ILP in a high-performance processor is typically achieved by aggressive speculation and out-of-order execution. These techniques increase the instructions per clock cycle but come at the cost of wasted work and wasted resources on never-committed instructions. The components, necessary for performing dynamic scheduling and out-of-order execution, account for a huge portion of the power budget [14]. Also, there is a limit in the amount of ILP that can be extracted. A point of diminishing returns has been reached, where the cost of extracting additional ILP is not in proportion to the resulting increase in performance.

Thread level parallelism (TLP) achieves a high throughput by execution of multiple threads and does not rely much on speculation. There are many applications where a high TLP can be exploited [15]. Hence, in order to increase processor performance in the future, an increased emphasis on thread-level parallelism is required. As technology improves and feature sizes of transistors are reduced, the switching speeds of the transistors increase. However, as discussed in the previous section, DSM issues such as interconnect delays, inductive effects, noise, crosstalk, and clock skew will be more manageable at the module level than at the global level. This will motivate designers to explore modular architectures that avoid long on-chip interconnect wires.

A promising solution is to develop single-chip parallel computers. Single-chip parallel computers are explicitly parallel architectures with multiple processing nodes on a single chip. It is projected that a system with 64 processors and 512 MB of memory can be built on a single chip within the next 8-10 years [16]. The length of the interconnect wires will depend on the size and complexity of the individual processors, not the size of the entire chip. Hence, long global interconnect wires can be eliminated by having simple processing cores. The ability to execute multiple threads on multiple processors allows the system to take advantage of TLP. This can lead to huge improvements in throughput,

especially in applications with high TLP. The modular architectures of these parallel systems can lead to significant improvements in performance without exponential increases in design complexity and design costs. Some research projects exploring single chip parallel computers include Hydra [17], TRIPS [18], Blue Gene [19], RAW [20] and the Single-Chip Message-Passing Parallel Computer (SCMP) [21].

The SCMP computer system has a modular architecture that can efficiently support TLP while minimizing the length of on-chip interconnect wires. The SCMP system includes up to 64 processors on a single chip, connected in a 2-D mesh with nearest neighbor connections. Memory is included on-chip with the processors and the architecture includes hardware support for communication and the execution of parallel threads. There are no global signals or shared resources. Avoiding long interconnect wires will allow the use of very high clock frequencies, which, when coupled with the use of multiple processors, will offer tremendous computational power. The absence of global signals, shared resources and aggressive speculation combined with other power optimizations render a power-efficient high-performance computer system.

1.4. Related Work

Recently, there has been a significant amount of research work focused on architectural level analysis and optimizations. David Brooks et al. have developed an architectural level power analysis and optimization tool called *Wattch* [22]. *Wattch* consists of parameterizable power models for different hardware structures integrated into the *SimpleScalar* simulator, *sim-outorder* [23]. Power dissipation is estimated using resource usage counts obtained through cycle-level simulation. *SimplePower* [24] developed by W. Ye et al. is also a cycle accurate energy estimation tool based on the *SimpleScalar* tool.

Leakage power dissipation is a major issue in DSM technologies and hence must be included in power estimation and analysis. *Wattch* and *SimplePower* do not account for leakage power dissipation. *HotLeakage* [25] is a leakage power estimation tool based on the *SimpleScalar* simulator, *sim-outorder*. The *SimpleScalar* tool set [23] is suited to

single-threaded single-processor architectures. So, power estimation tools based on the SimpleScalar tool cannot be used for multiprocessor architectures. On-chip interconnect networks are major contributors to power dissipation in single chip parallel computers. *Orion* [26], developed by Hang-Sheng Wang et al., is a power-performance simulator for interconnection networks. Orion, however, does not model power dissipation in the processor nodes. Single Chip Parallel Computers need power estimation tools that estimate dynamic and leakage power dissipation in the processor nodes and the on-chip interconnect network.

1.5. Thesis Organization

This thesis focuses on the development of a power analysis and optimization tool for Single Chip Parallel Computers. Chapter 2 presents the background for power estimation in CMOS Circuits. The methodology used in this work to estimate dynamic and static power dissipation at the architectural level is presented. The architecture of the SCMP system is briefly described. Chapter 3 provides a detailed description of the development of power models for dynamic and static power dissipation in the processor cores and the on-chip network. Power modeling is based on the SCMP architecture but the approach can be easily generalized and applied to other architectures. Chapter 4 describes the integration of the power models into a detail microarchitectural performance simulator for the SCMP system. The different options and configurations available in the power and performance analysis tool are presented. Chapter 5 presents the results of power and performance analysis of the SCMP system using the developed tool. The power advantages identified in the SCMP architecture are described. Power optimizations are presented and the design trade-offs involved in the design of a single chip parallel system like SCMP are discussed. Chapter 6 presents the conclusions of this work and some proposals for future research.

Chapter 2

2. Background

Architectural Level Power analysis and optimization in a Single Chip Parallel computer requires microarchitectural power models. In order to construct the power models it is important to have an in-depth understanding of power dissipation in CMOS circuits and architectural level power estimation methodologies. This chapter covers the basics of power dissipation in CMOS circuits. The architectural level power estimation methodologies for dynamic and static power estimation, followed in this work, are presented. An introduction to the SCMP architecture is provided for easy understanding of the development of the power analysis tool and the proposed power optimizations.

2.1. Power Dissipation in CMOS circuits

Power dissipation in digital CMOS circuits can be classified into two types: dynamic power dissipation and static power dissipation. Dynamic power dissipation is due to high-to-low and low-to-high signal switching in circuits. Static power dissipation depends on the logic states of the circuit. It does not depend on signal switching [27]. The average power dissipation in a digital CMOS circuit can be given by the following equation [4]:

$$P_{avg} = P_{sw} + P_{sc} + P_{leak} + P_{static} \quad (2.1)$$

where, P_{sw} is the capacitive switching power dissipation, P_{sc} is the short-circuit power dissipation, P_{leak} is the power dissipation due to leakage currents and P_{static} is the static power dissipation due to non-leakage static currents. Capacitive switching power and short-circuit power are components of dynamic power dissipation. Leakage power is a major component of static power dissipation in CMOS circuits, though there might be

some non-leakage currents that contribute to a small percentage of static power dissipation.

2.1.1. Capacitive Switching Power Dissipation

Dynamic power dissipation is the most significant source of power dissipation in digital CMOS circuits and capacitive switching power is the largest contributor to dynamic power dissipation. Capacitive switching power is the power dissipation caused by charging and discharging of capacitances. In digital CMOS circuits, capacitances are formed from parasitics in the transistors and interconnection wires. These parasitic capacitances cannot be eliminated. Their estimation is crucial for dynamic power analysis and optimizations.

The derivation of capacitive switching power can be understood with the example of a CMOS inverter as shown in Figure 2.1. The inverter drives a load capacitance C_L . C_L is the equivalent capacitance representing the cumulative effect due to the parasitic capacitances associated with the nMOS and pMOS transistors, the parasitic capacitances due to interconnect wires at the inverter's output and the input capacitances of the gates driven by the inverter. Consider the inverter in Figure 2.1. Assume that initially it is in steady state with input at logic HIGH and output at logic LOW. If there is a falling transition at the input from HIGH to LOW as shown in Figure 2.2(a), then the nMOS transistor turns off while the pMOS transistor turns on. The capacitance, C_L , is charged to V_{dd} . Energy equal to $C_L \cdot V_{dd}^2$ is drawn from the power supply. One half of this energy is dissipated across the pMOS transistor and the interconnect while the other half is stored in the capacitance C_L . If the input undergoes a rising transition from LOW to HIGH as shown in Figure 2.2(b), then the pMOS transistor turns off while the nMOS transistor turns on. The capacitance C_L is completely discharged. The energy stored in the capacitor gets dissipated across the nMOS and the interconnect.

If the capacitance C_L is charged and discharged α times during a time period $[0, T]$, the power dissipated in CMOS inverter can be given by the following equation.

$$P_{sw} = C_L \cdot V_{dd}^2 \cdot \alpha \cdot (1/T) \quad (2.2)$$

The following assumptions are made in the derivation [27]:

- i) The capacitance C_L remains constant.
- ii) The supply voltage V_{dd} is constant.
- iii) The capacitance C_L is fully charged to V_{dd} and fully discharged to 0 V .

In properly designed digital CMOS circuits, these assumptions are satisfied and the above equation estimates capacitive switching power accurately. If the inverter is a part of a synchronous circuit running at a clock frequency f , then the power dissipation in the CMOS inverter can be given by the following equation:

$$P_{sw} = C_L \cdot V_{dd}^2 \cdot \alpha \cdot f \quad (2.3)$$

The variable α is known as the activity factor as it quantifies the switching activity in the circuit. The activity factor in equation (2.3) specifies the number of times the capacitor charges and discharges in a given clock period. In an ideal synchronous circuit running at a clock frequency f the maximum rate of change of any signal will be f . Hence, the activity factor α will be a fraction between 0 and 1. The factor $\alpha \cdot f$ is equal to the rate at which the capacitor C_L charges and discharges.

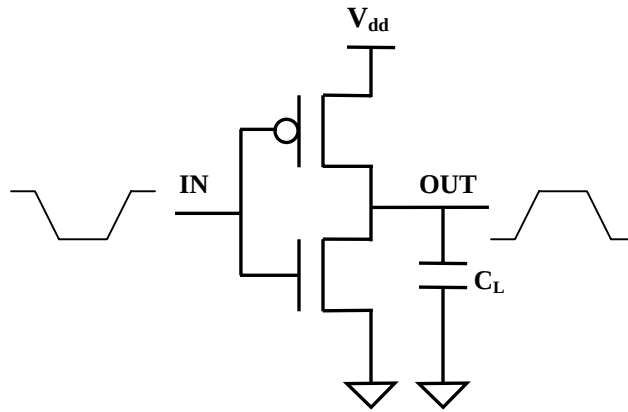


Figure 2.1: CMOS Inverter

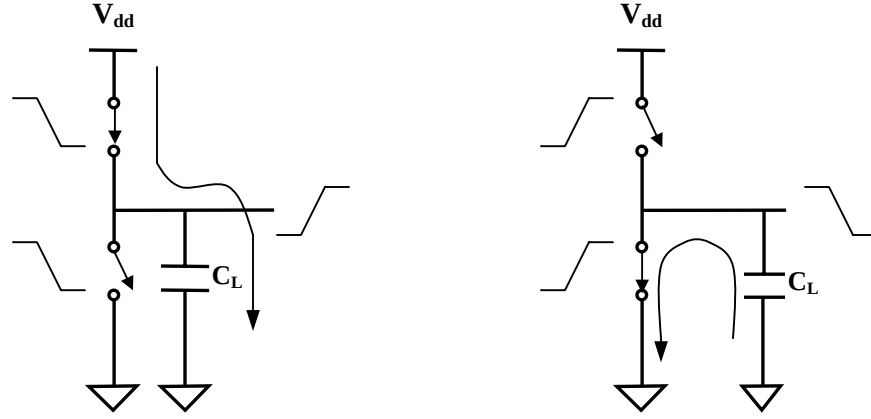


Figure 2.2: Equivalent Circuit for: (a) Charging Capacitor (b) Discharging Capacitor

2.1.2. Short Circuit Power Dissipation

Short circuit power is a component of dynamic power dissipation in CMOS circuits. It is caused by the flow of short circuit current between supply and ground during switching or transition in signal values [27]. Consider the inverter in Figure 2.1. Whenever there is a transition in the input signal, there is short duration in which the input signal is between the threshold voltages of the nMOS and the pMOS transistors and both transistors are turned on. This causes short circuit current to flow from supply to ground and results in short circuit power dissipation.

If symmetric fall and rise times and threshold voltages are assumed, the short circuit power dissipation in a CMOS inverter can be approximately given by the following equation [28].

$$P_{sc} = K. (V_{dd} - 2V_T)^3. \tau. N. f \quad (2.4)$$

where, K is a constant that depends on the transistor sizes and the technology, V_T is the magnitude of the threshold voltage of the nMOS and pMOS transistors, τ is the input rise/fall time, N is number of transitions at inverter's output and f is the clock frequency. Note that $N = 2. \alpha$, where α is the activity factor.

Proper sizing of transistors and reduction in rise and fall times control short circuit power dissipation. Reducing the switching activity in the circuit reduces short circuit power dissipation. Short circuit power dissipation is only a small percentage of dynamic power dissipation in CMOS circuits. Optimizations that reduce switching activity in the circuit reduce both capacitive switching power and short circuit power.

2.1.3. Leakage Power Dissipation

Leakage power dissipation is a component of static power dissipation in CMOS circuits. It is caused by the presence of leakage currents in the MOS transistors. The major sources of leakage current in CMOS circuits are (Figure 2.3) [29]:

- i) Subthreshold channel conduction current.
- ii) Gate Direct Tunneling Current
- iii) Reverse Biased PN-junction current.

When a transistor is logically turned off, a non-zero leakage current flows through the channel. This happens when the gate voltage is below the threshold voltage. Hence, this leakage current is known as subthreshold leakage. Subthreshold leakage increases significantly with technology scaling and hence is a concern in DSM designs. When the operating voltage is reduced, the threshold voltage must be reduced to compensate for the increase in delay. Reduction in threshold voltage drastically increases subthreshold leakage [4].

Gate direct tunneling current occurs from tunneling of electrons or holes from the bulk and source or drain overlap region through the gate oxide potential barrier into the gate or vice-versa [29]. The International Technology Roadmap for Semiconductors (ITRS) [13] projects that the gate oxide thickness will scale aggressively for future technology nodes in an effort to improve device performance. The tunneling current increases exponentially with the decrease in the oxide thickness and with increase in the potential drop across oxide. Hence, gate leakage through the gate oxide will increase significantly with technology scaling due to the direct tunneling current and will be dominant for the 70nm and smaller technology nodes [25].

The third source of leakage current occurs from the reverse biased PN junction formed at the source or drain of transistors due to parasitic effect of the bulk of CMOS device structure. The reversed biased PN junction is formed when the source or drain of the nMOS (pMOS) is at V_{dd} (Gnd). The junction current is picked through the bulk or well contact. The current depends on the temperature, bias voltage, area of the junction and the fabrication process.

The subthreshold current, which is the most significant source of leakage current, increases exponentially with the temperature. The gate leakage is almost independent of the temperature variation while the junction tunneling leakage increases slowly with temperature. The sensitivity of leakage power dissipation to temperature variation can lead to thermal runaway [30]. Leakage power dissipation increases with temperature and the increase in leakage power dissipation can further increase the temperature. If the heat removing ability of the package is adequate, equilibrium between increases in temperature and removal of heat will be reached. In case when the heat removal ability is not adequate, the temperature and leakage power will interact in a positive feedback and increase each other leading to a thermal runaway.

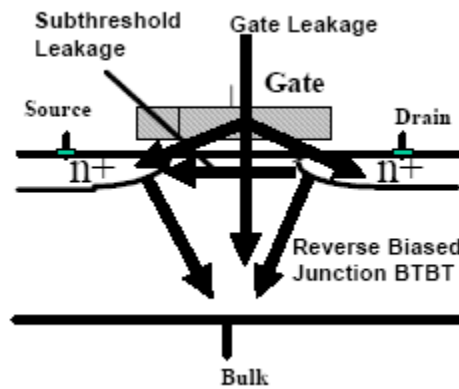


Figure 2.3: Sources of Leakage in a MOSFET

2.1.4. Static Power Dissipation

Ideally, leakage power dissipation is the only source of static power dissipation in fully complementary digital CMOS circuits. In properly designed circuits non-leakage static currents can be avoided. Static power dissipation can however result from degenerated voltage levels at the inputs to static gates [4]. Bus contention and signal conflicts due to multiple drivers also result in static power dissipation.

2.2. Architectural Level Power Estimation

In architectural level power estimation, lower level details are abstracted and accuracy is sacrificed for increases in simulation speed. However, reasonable accuracy is sufficient for power analysis at the architectural level. Architectural and microarchitectural level power optimization and analysis rely on relative accuracy rather than absolute accuracy. Power estimation in CMOS circuits consists of estimation of dynamic power dissipation and estimation of static power dissipation. The power dissipated in a single chip parallel computer can be estimated by estimating the power dissipated in the processor nodes and the power dissipated in the on-chip interconnect network. This section details the architectural level power estimation strategies used in this work.

2.2.1. Estimation of Dynamic Power Dissipation

The most significant source of dynamic power dissipation in CMOS circuits is capacitive switching. Only a small percentage of dynamic power dissipation is caused due to short circuit currents. Hence, short circuit power dissipation can be ignored for architectural level analysis and optimization. As shown in Section 2.2.1, capacitive switching power dissipation is expressed as $P = C.V_{dd}^2.\alpha.f$, where C is the switching capacitance, V_{dd} is the supply voltage, α is the activity factor and f is the clock frequency. The switching capacitance of a hardware component depends on the process technology and the hardware implementation of the component. If the implementation details and the

process technology are specified for the component, its switching capacitance can be estimated. The clock frequency and supply voltage can be specified for a given simulation run. The activity factor, expressed as a fraction between 0 and 1, is estimated based on the switching activity during each clock cycle.

The infrastructure used to estimate dynamic power dissipation in this work is based on the architectural level power analysis tool *Wattch* [22]. In order to estimate the power dissipation of a system, parameterizable power models are constructed for each of the hardware components constituting the system. The power models take inputs such as process technology, supply voltage, clock frequency and design configurations. The models estimate the switching capacitance and the factor $C.V_{dd}^2.f$ in the power dissipation equation. The activity factor α is then estimated based on the number of times the hardware component is accessed during each clock cycle. The access counts are obtained by executing benchmarks using a microarchitectural simulator.

The power models basically model the switching capacitance of the component based on the specified design configuration and process technology. The architectural simulator *Wattch* was shown to have reasonable accuracy in modeling the capacitance of the functional units in a processor. Hence, the methodology used to estimate the capacitance of basic hardware units, in the processor nodes, is based on that followed in *Wattch*. The interconnect network in a single chip parallel computer is a significant source of power dissipation. The basic procedure followed to estimate the capacitive switching power in the interconnect network is similar to that followed to estimate the same in the processor nodes. The capacitance models for the network subsystems are based on the models used in the network power-performance simulator, *Orion* [26].

2.2.2. Estimation of Static Power Dissipation

Dynamic power dissipation is the major source of power dissipation in current processors. However, technology scaling is increasing the contribution of static power dissipation and its significance at the system level. The ITRS [13] predicts that in future, leakage power will become a dominant source of power dissipation in processors. A considerable amount of research is focused on reducing static power dissipation at the architectural level [29, 30]. Microarchitectural level leakage modeling is essential for

exploring architectural level leakage power optimizations. The leakage models must be parameterizable, accounting for all the important parameters that effect leakage and must be integrable into a microarchitectural simulator.

Butts and Sohi proposed a leakage model [31] that is simple and can easily be used for architectural static power estimation. However, the Butts and Sohi model does not explicitly consider temperature scaling and is not suitable for simulation studies [25,30]. In this work the leakage models are based on the temperature aware leakage model, *HotLeakage* [25]. The Butts and Sohi model omits some factors affecting leakage that are now being considered important. But the framework of the model makes architectural level leakage estimation easy and efficient [25]. The models in *HotLeakage* and hence the models in this work adapt the leakage modeling structure from the Butts and Sohi model.

The leakage models take into account subthreshold leakage and gate leakage. The subthreshold leakage models are based on the BSIM3 v3.2 [32] leakage current equation in a MOSFET transistor. Lower level model parameters are obtained through transistor level simulations and are stored in look-up tables. Gate leakage is modeled using the BSIM4 [33] technology data and equations. Once the transistor level leakage is modeled, the leakage for each basic hardware cell (for example a SRAM cell) is calculated. The models are integrated into a microarchitectural simulator and the leakage power dissipation is estimated based on the configuration provided by the user for every simulation run. In this work, the leakage power dissipation in the interconnect network of the single chip parallel computer is ignored. Leakage models are developed for important functional units in the processor nodes.

2.3. SCMP Architecture

The SCMP system has a tiled architecture [21], with up to 64 simple RISC processors and a two-dimensional mesh network through which nodes communicate as shown in Figure 2.4. Each tile or each processing node contains a CPU, memory, and networking components. Figure 2.5 shows a block diagram for a SCMP tile.

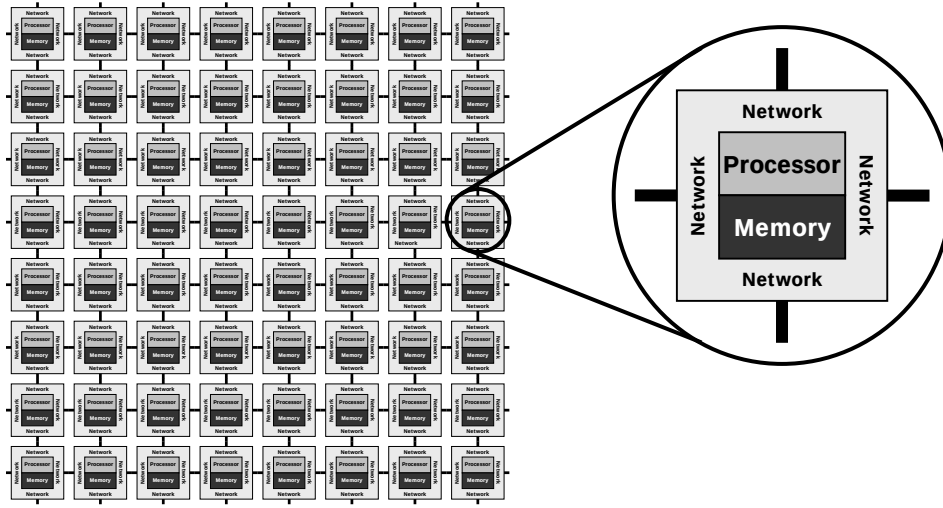


Figure 2.4: SCMP system with 64 tiles; each having a CPU, memory & network interface

2.3.1. CPU Architecture

The SCMP CPU is a RISC core with custom features enabling message passing and multiple threads of execution. The CPU is controlled through a 4-stage in-order pipeline made up of the conventional Fetch, Decode, Execute and Write Back stages. The processor includes an integer ALU and a floating point ALU. It supports single and double precision floating point operations. Having an on-chip local memory next to each CPU eliminates the need for a data cache and the added complexity that comes with it. The SCMP CPU is expected to scale well with clock frequency due to the short interconnect wires and embedded local memory.

Each processor includes hardware support for up to 16 threads, with hardware-based non-preemptive round-robin scheduling. The processors use multiple threads to overlap communication latency with computation by performing a context switch when a thread is waiting for message data. Each thread has its own associated context. Each processor's execution threads are created, managed, and scheduled using a Context Management Table, or CMT. A CMT control logic module interfaces the CMT to the pipeline, or NIU, and also the memory controller. It updates the CMT based on inputs from the

pipeline, the memory controller or NIU and also provides the desired data from the CMT to the pipeline or memory controller.

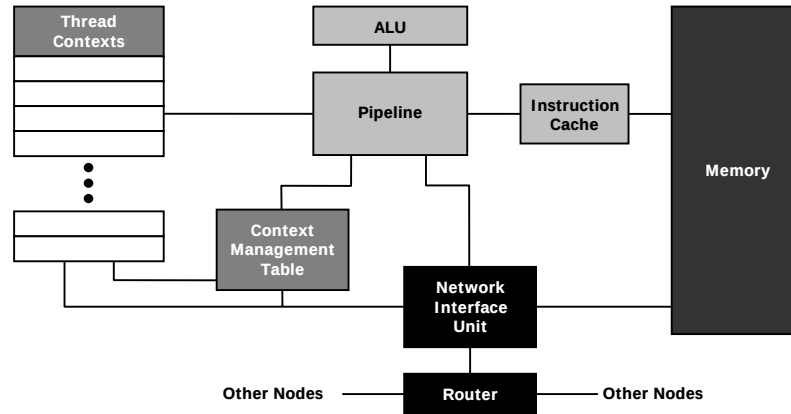


Figure 2.5: Block Diagram of a SCMP tile.

2.3.2. Memory Subsystem

Each processor node in the SCMP system will include between 1 and 8 MB of embedded DRAM. The processors have direct access only to their local memory. Data is shared among processors through communication. The CPU should be able to access any location of the local memory in one or two clock cycles. Therefore, no data caches are needed in the SCMP system. A 32KB instruction cache is included, however, to prevent interference when the pipeline is reading or writing to memory. A memory controller is implemented to provide glue logic between the memory and rest of the functional units in the processor node.

2.3.3. Network architecture

The SCMP message-passing network and the processor-network interface are designed to minimize overhead and reduce communication latency for supporting efficient fine-grain and medium-grain parallelism. The message passing is based on an

active-message style, similar to that used in the J-Machine and Pica architectures [34, 35]. Messages are broken into flow-control digits, or flits, representing individual pieces of data that are atomically transmitted through the network.

The processors are connected in a 2-D mesh. Each processor has a direct connection to its four nearest neighbors (North, South, East, and West), and dimension-order wormhole routing is used for communication. In each node, the networking subsystem consists of a Network Interface Unit (NIU) and a router. The NIU is responsible for collecting data from the local node to be sent across the network and for receiving messages from the network and distributing the data among the other local subsystems.

Since each node is connected to only its four nearest neighbors, wire lengths will remain very short, allowing high-bandwidth, low-latency network connections. In addition, the routers communicate with each other using an asynchronous handshaking protocol, which eliminates the need for a synchronized global clock signal and the associated clock skew problems. This will enable SCMP chips to scale well with future fabrication technology and application performance demands.

2.3.4. Hardware Implementation

The SCMP system can be implemented with up to 64 processor nodes. Each processor node has a CPU core and an embedded DRAM local memory. We estimate the CPU core to have a transistor count of not more than 2.5 million transistors, including the ICache and the register contexts. The DRAM local memory can be 1 MB to 8 MB in size, depending on the applications the SCMP is intended to support. Based on the ITRS Roadmap [13], we predict that by the year 2006, using the 70 nm technology node, it would be possible to integrate 64 processor nodes with 1 MB of DRAM each or 32 processors with 4 MB of DRAM each on a single chip. By 2012, using the 35nm technology node, 64 processor nodes with 8 MB local memory could be efficiently integrated on a single chip

Chapter 3

3. Power Models for SCMP

In microarchitectural level power estimation and analysis, lower level details and accuracy are sacrificed for increase in simulation speed and portability. However, power optimization and analysis at higher levels of abstraction are based on relative accuracy. So, relative accuracy is more important than absolute accuracy at the architectural level. The construction of the dynamic and static power models for the SCMP architecture is explained in this chapter. The approach followed to construct the microarchitectural power models for the processor nodes and the network can be generalized and applied to other similar architectures.

3.1. Dynamic Power Models for the CPU

Capacitive switching power is the largest contributor to dynamic power dissipation. As discussed in Section 2.1 capacitive switching power is the power dissipated due to charging and discharging of capacitances. These capacitances are formed from parasitics in the transistors and interconnection wires. In order to estimate the capacitive switching power dissipation the parasitic capacitances have to be estimated. The switching capacitance in a component depends on the technology node and the hardware implementation of the component. In *Wattch* [22], the capacitance models are constructed for all the components for a technology node (0.35 technology node) and then scaling factors are applied to the capacitance models in order to scale them to other technology nodes. A similar approach is followed in this work.

The first step in estimating the dynamic power dissipation in the CPU or processor node is to divide the CPU into smaller components and then estimate switching capacitance for each of the components. The switching capacitance of a component depends on the internal capacitances and the implementation of the component. The

assumptions made for estimating the internal capacitances are similar to those in *Wattch*. The construction of dynamic power models is explained for some of the major components in the SCMP processor node.

3.1.1. Dynamic Power Model for the Register Contexts

The register contexts are basically SRAM arrays. The power model for the register contexts is parameterized based on the number of contexts, the number of registers per context and the number of read and write ports. By default sixteen contexts are assumed with thirty-two 32-bit registers per context. The default number of ports is two read ports and one write port. The power dissipated in a context is given by the following equation.

$$P_{context} = P_{decoder} + P_{bitline} + P_{wordline} + P_{sense} + P_{cell} \quad (3.1)$$

where, $P_{decoder}$ is the power dissipated in the array decoder, $P_{bitline}$ is power dissipated in the array bitline, $P_{wordline}$ is the power dissipated in the array wordline, P_{sense} is the power dissipated in the sense amplifiers and drivers and P_{cell} is the power dissipated in the SRAM cell. A 6-T SRAM is assumed as shown in Figure 3.1. The array decoder, bit line, word line and sense amplifiers are the major sources of power dissipation in an SRAM array. The power dissipated in these components is obtained from their respective power models which in-turn have to be parameterized with parameters such as number of columns, number of rows, number of ports, bitline length and wordline length. The bitline and wordline lengths are determined from number of rows, number of columns, number of ports, RAM cell height and wordline spacing. The assumptions for RAM cell height and wordline spacing are similar to those in *Wattch*.

The power models for the various components in the SRAM array are also used for other array components in the CPU. Each of the models estimate the power by estimating the switching capacitance involved. For example, the capacitance in each wordline and the capacitance in each bitline can be estimated using the equations in Table 3.1 [22].

$$WordlineCapacitance = C_{diff}(WordLineDriver) + C_{gate}(CellAccess) \times NumberBitLines + C_{metal} \times WordLineLength \quad (3.2)$$

$$Bitline\ Capacitance = C_{diff}(Pre-charge) + C_{diff}(CellAccess) \times NumberWordLines + C_{metal} \times BitLineLength \quad (3.3)$$

Table 3.1: Wordline and Bitline Switching Capacitance

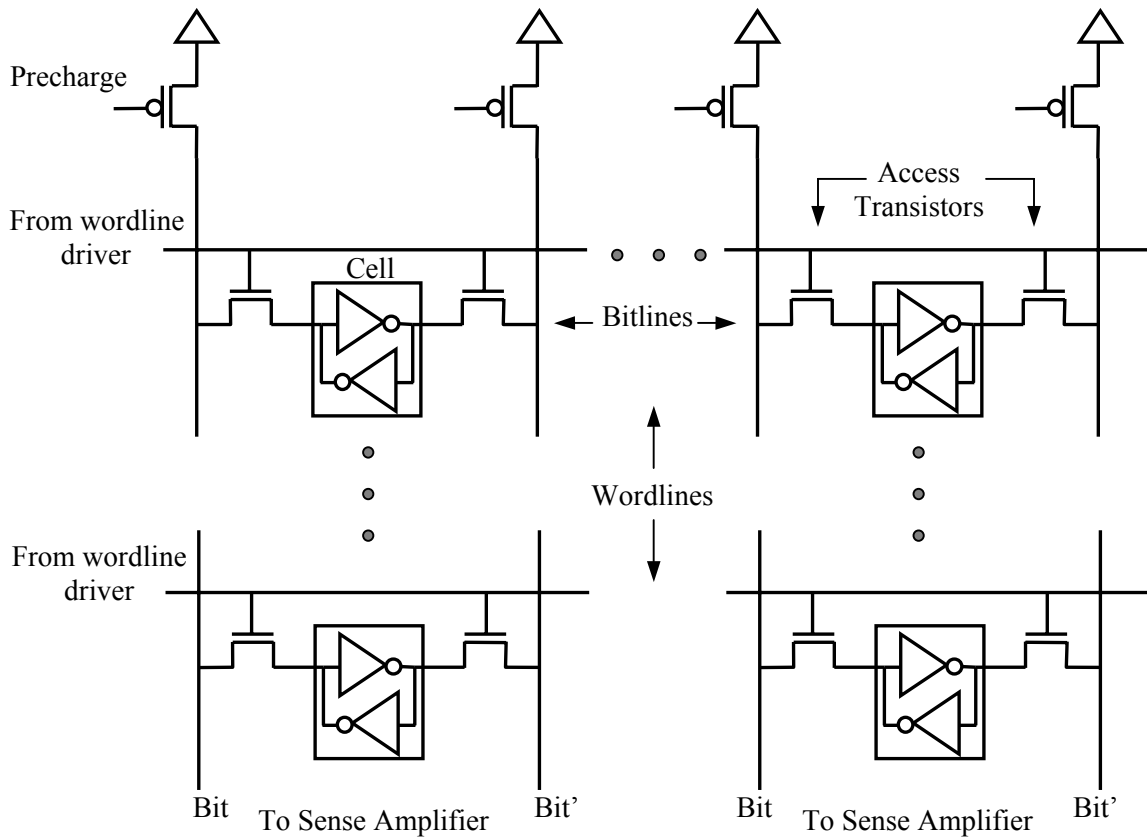


Figure 3.1: Wordlines and Bitlines in an SRAM Array Structure

In words, the switching capacitance in the wordline is the cumulative sum of the diffusion capacitance of the wordline driver ($C_{diff} (WordLineDriver)$), the gate capacitance of the cell access transistor times the number of bitlines ($C_{gate} (CellAccess) * NumberBitLines$) and the metal capacitance of the wordline ($C_{metal} * WordLineLength$). The transistor sizing strategy is adopted from *Wattch*. Similarly, the switching capacitance in the bitline is given by the cumulative sum of the diffusion capacitance of the pre-charge transistor, number of wordlines times the diffusion capacitance of the cell-access transistor and the metal capacitance of the bitline.

The gate capacitance in a CMOS transistor can be estimated by the following equation [36].

$$C_g = C_{ox} \times W \times L + W \times C_{gso} + W \times C_{gdo} + (2L \times C_{gbo}) \quad (3.4)$$

where W is the width of the gate channel, L is the length of the gate channel, C_{gso} , C_{gdo} , and C_{gbo} are the capacitances formed from fringing fields and overlapping materials. C_{ox} is the capacitance of the thin-oxide, which is given by the following equation.

$$C_{ox} = (\epsilon_0 \cdot \epsilon_{SiO2}) / t_{ox} \quad (3.5)$$

where, the ϵ_0 is the permittivity of free space equal to 8.854×10^{-14} , ϵ_{SiO2} is the relative permittivity of SiO_2 equal to 3.9 and t_{ox} is the thickness of the thin-oxide, obtained from SPICE model parameters. The diffusion capacitance can be estimated from the following equation [36].

$$C_d = C_j \times ab + C_{jsw} \times (2a + 2b) \quad (3.6)$$

where, a and b are the width and height of the diffusion region, respectively, C_j is the junction area capacitance, and C_{jsw} is the capacitance of sidewalls of the diffusion region. C_j and C_{jsw} are both obtained from SPICE models.

The array decoder and sense-amplifier switching capacitances are estimated in a similar manner. Once the switching capacitance is estimated from the given parameters, it is multiplied by the technology scaling factor, the supply voltage and the frequency. The supply voltage and frequency are provided at the start of the simulation while the technology-scaling factor is determined from the selected technology node. The activity-factors are estimated from the total number of context read and write accesses, which in turn are determined through the execution of benchmarks in the microarchitectural simulator.

3.1.2. Dynamic Power Model for the CMT & CMT Control

The Context Management Table, or CMT is a table or array that contains the information for each of the register contexts in a node. Each entry in the CMT indicates whether the associated context is in use, whether it contains an active thread, as well as the address of the next instruction to be executed by the thread. The CMT Control logic maintains and updates the data in the CMT. It monitors inputs and commands from the NIU, the pipeline and the memory controller. In the power simulator, power models for some special registers are also included under the CMT power model. So, the CMT power model consists of the power models for the CMT array, the CMT Control logic and the special registers.

The power model for the CMT array and the special registers will be similar to that of the register contexts. The CMT Control logic decodes the commands from the NIU, the Pipeline and the Memory Controller and generates the read and write control signals for the CMT array. The Pipeline, Memory Controller or the NIU can update the CMT while the Pipeline and Memory Controller can read a CMT entry. A CMT entry corresponding to a context consists of several subentries. The commands from the NIU, the Memory Controller and Pipeline are for updating or reading only these subentries. Hence, masks are generated for the read or write operation based on the output of the decoding logic.

The access to the CMT array is prioritized. The Pipeline is given the highest priority while the NIU is given the least. So, priority encoded access logic is used. The control is established using a Finite State Machine (FSM). Hence, the state registers and the combinational logic must be included in the power model. Implementation level details

for the logic required in the CMT Control are considered in terms of basic blocks. The power in the CMT control logic can be given by the following equation.

$$P_{cmt_ctrl} = P_{decoder} + P_{fsm} + P_{priority} + P_{mask} + P_{mux} \quad (3.7)$$

where, $P_{decoder}$ is the power dissipated in the decoders, P_{fsm} is the power dissipated in the control FSM, $P_{priority}$ is power dissipated in the priority encoding logic, P_{mask} is power dissipated in the mask generation logic, P_{mux} is power dissipated in the multiplexers. All these components of power dissipation are obtained from power models that can be parameterized. Consider the power model of the decoding logic. It again consists of three components:

- 1) Decoder for pipeline commands: 3 to 8 decoder
- 2) Decoder for memory commands: 3 to 8 decoder
- 3) Decoder for NIU commands: 2 to 4 decoder

The selection of the decoder size is based on the number of commands to be decoded. For each of the above components, a power model for a decoder that can be parameterized with the number of inputs or outputs is used. The power model for a ‘ n to k ’ decoder shown in Figure 3.2 can be constructed based on the estimation of the switching capacitance given by the following equations.

$$Decoder\ Capacitance = C_{input} \times n + C_{output} \times k \quad (3.8)$$

$$C_{input} = 2 \times C_{diff}(InputDriver) + C_{gate}(DecoderNand) \times k \quad (3.9)$$

$$C_{output} = C_{diff}(DecoderNand) + C_{gate}(OutputInverter) \quad (3.10)$$

Once the switching capacitance is estimated, it is multiplied to the supply voltage, frequency and the technology-scaling factor. The activity factor is obtained from the microarchitectural simulator. The power model for the decoder can be parameterized using either the number of inputs, n ($k = 2^n$) or the number of outputs, k ($n = \log_2(k)$).

Note that it is assumed that decoders up to dimensions of ‘3 to 8’ are implemented as shown in Figure 3.2 and decoders of dimensions higher than ‘3 to 8’ are constructed from decoders of dimensions equal to or smaller than ‘3 to 8’. This assumption is made because the decoder implementation in Figure 3.2 will require impractically high fan-in gates for decoders of higher dimensions.

The power models for other basic components can also be constructed in a similar fashion. The power model for a FSM can be parameterized with the number of states and the number of inputs. The power model for the priority encoding logic and the multiplexers can be parameterized using the number of inputs or outputs. All major modules in the system can be implemented using some basic components. Hence, parameterizable power models for basic blocks can be constructed and used across most of the system modules. However, some components in a module have to be uniquely modeled. For example, the power model for the mask generation logic is uniquely constructed for the CMT Control logic.

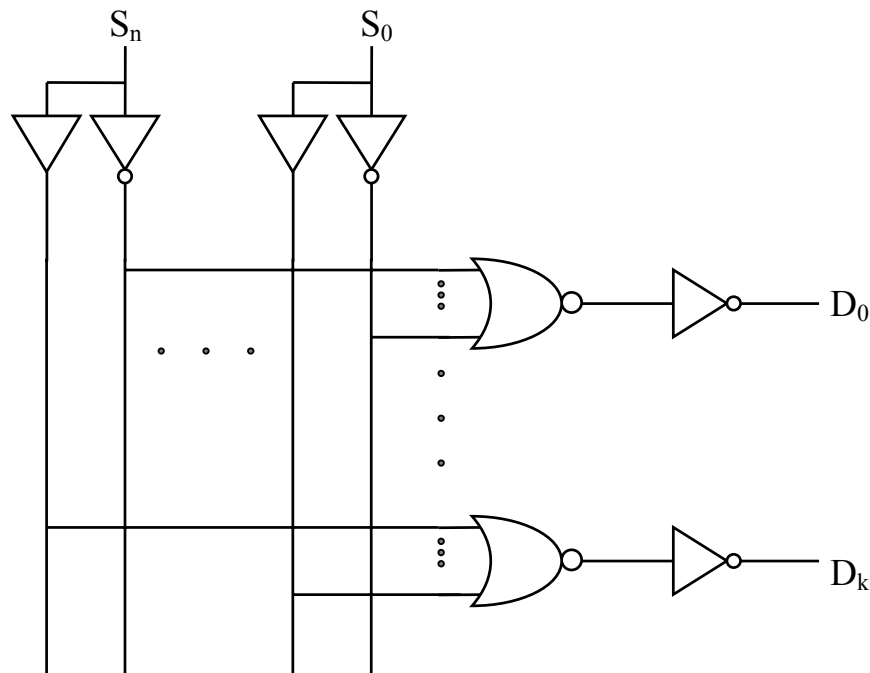


Figure 3.2: A ‘ n to k ’ Decoder

3.1.3. Dynamic Power Models for other components

The basic approach followed in constructing the power model for any hardware module is first considering the hardware implementation of the module in terms of basic components like array structures, registers, control FSM, decoders, multiplexers etc. Then using parameterizable power models for each of the basic components to estimate the switching capacitance of the hardware module. The final step in determining the power dissipation is the estimation of the activity factor through simulation. This basic approach is followed for the Register Contexts and the CMT Control logic. Power models for all the modules in SCMP are also constructed in a similar fashion.

The power models for the Integer ALU and the Floating Point ALU are constructed using power numbers from previous work [37, 38] that are scaled for process technology, frequency and supply voltage. The power numbers estimate the power dissipation for every ALU operation. The power dissipation is taken into account whenever an operation is executed. This information about the execution of operations is available dynamically through microarchitectural simulation. The Instruction Cache (ICache) and the Memory are array structures and the approach to construct their power models, is similar to that followed for the Register Contexts. The Memory Controller enables the Pipeline, the ICache and the NIU to have priority-based access to the memory. The Memory Controller will have a control FSM, a decoding logic (to decode the commands from the Pipeline, ICache and NIU), a priority encoder and multiplexers. So, the power model for the Memory Controller can be constructed using parameterizable models of the basic components.

The Pipeline consists of Finite State Machines, pipeline registers, decoding logic, comparators, address generation logic and forwarding logic. The address generation logic consists of adders and multiplexers while the forwarding logic is a set of multiplexers. Hence, the power model for the pipeline can be constructed using parameterizable power models of the basic components. The implementation of the NIU consists of the logic that injects flits into the network and the logic that ejects flits from the network. The NIU power model is divided into the Inject power model and Eject power model. The NIU power model mainly consists of the control Finite State Machines, multiplexers, flit construction and distribution logic and First-In-First-Out buffers. The capacitance model

for the flit construction and distribution logic must be uniquely constructed for the NIU. All other switching capacitances can be obtained from parameterizable models of the basic components.

3.2. Dynamic Power Models for the Network

The power models for dynamic power dissipation in the network are based on *Orion*, a power-performance simulator for interconnection networks [26]. The power modeling strategy for the interconnection network is similar to that followed for the components in the CPU. In SCMP, dimension-order wormhole routing is used for communication. The block diagram of a wormhole router is shown in Figure 3.3 [39]. The power model for the router is constructed by decomposing it into basic components: memory arrays, crossbars and arbiters. The total power dissipated in the router shown in Figure 3.3, can be given by:

$$P_{router} = 5 \times P_{buffer} + P_{crossbar} + 5 \times P_{arbiter} \quad (3.11)$$

where, P_{buffer} is the power dissipated in a flit buffer, $P_{crossbar}$ is the power dissipated in the crossbar and $P_{arbiter}$ is the power dissipated in an arbiter.

3.2.1. Power Model for the Flit Buffers

In the SCMP system, messages are transferred between processor nodes in the form of flits or flow control units. The flits are buffered using FIFO SRAM channel buffers with independent read and write ports. The power consumption in the FIFO SRAM buffer is given in [39] by the following equations.

$$P_{buffer} = f_{clk} \times (5 \times P_f \times (E_{write} + E_{read})) \quad (3.12)$$

$$E_{write} = E_{wl} + F \times P_{fac} \times (E_{bw} + E_{cell}) \quad (3.13)$$

$$E_{read} = E_{wl} + F \times P_{fac} \times (E_{br} + 2 \times E_{chg} + E_{amp}) \quad (3.14)$$

where, f_{clk} is the clock frequency, F is flit width, P_f is the flit arrival rate, P_{fac} is the probability of signal changing value, E_{wl} is the energy dissipated in the wordlines, E_{bw} energy dissipated in bitlines during write, E_{br} energy dissipated in the bitlines during read, E_{cell} is the energy dissipated in the SRAM cell, E_{chg} is the energy dissipated in the pre-charge transistor and E_{amp} is the energy dissipated in the sense amplifiers and drivers. Note that: $E_x = C_x V_{dd}^2$ where, C_x is the corresponding switching capacitance. P_f and P_{fac} represent the activity factor when combined.

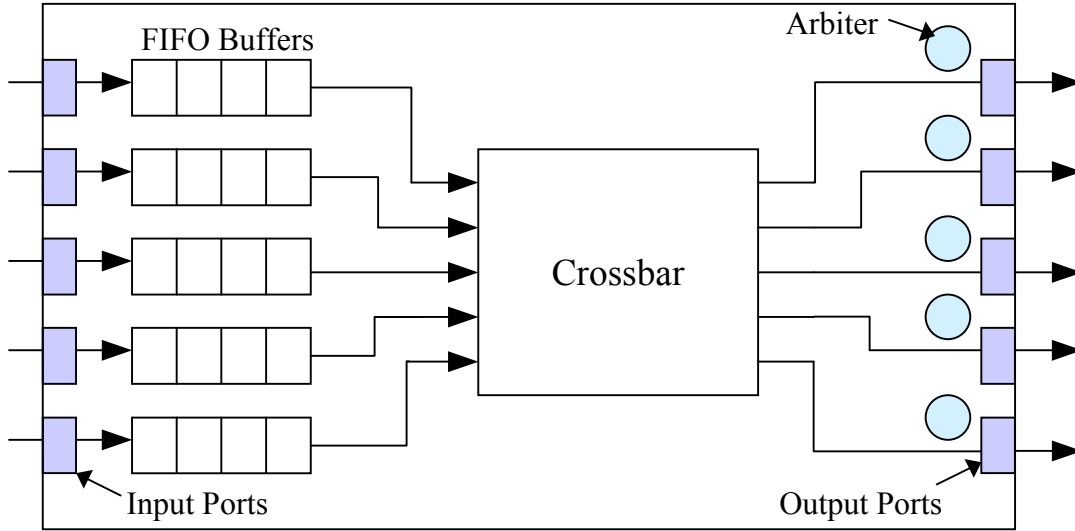


Figure 3.3: Wormhole Router

3.2.2. Power Model for the Crossbar

The crossbar implementation in on-chip interconnection routers can be of different types. The crossbar in the SCMP router is a Matrix Crossbar. The implementation of the SCMP 5×5 Crossbar switch is shown in Figure 3.4. The crossbar connects input paths to output paths enabling routing of flits. Connections determine which inputs are connected to which outputs. The data from the input port is present at the inputs of the connectors in that column. The open or close state of the connector determines which output ports

receive the data. In SCMP, dimension-order routing is used and hence all connections are not required. The flits move the X dimension first and then the Y dimension. So, connections violating the dimension-order are restricted. For example, a signal in the south direction cannot move in the east or west directions.

Power consumption in the 5×5 Crossbar can be estimated using the following equation developed in [40].

$$P_{crossbar} = f_{clk} \times (5 \times P_f \times P_{fac} \times F \times (E_{cb_in} + E_{cb_out})) \quad (3.15)$$

where, f_{clk} is the clock frequency, F is flit width, P_f is the flit arrival rate, P_{fac} is the probability of signal changing value, E_{cb_in} is the energy dissipated in the crossbar input lines and E_{cb_out} energy dissipated in crossbar output lines.

$$E_{cb_in} = C_{cb_in} \times V_{dd}^2 = (5 \times C_{in_cnt} + C_{d_in} + C_{wm1} \times L_{in}) \times V_{dd}^2 \quad (3.16)$$

$$E_{cb_out} = C_{cb_out} \times V_{dd}^2 = (5 \times C_{out_cnt} + C_{g_out} + C_{wm2} \times L_{out}) \times V_{dd}^2 \quad (3.17)$$

$$L_{in} = 5 \times W \times w_t \quad (3.18)$$

$$L_{out} = 5 \times W \times h_t \quad (3.19)$$

where, W is the port width ($F+I$ in SCMP), w_t is the width of input/output routing line, h_t is the height of the input/output routing lines, C_{in_cnt} is the capacitance of the connector input, C_{out_cnt} is the capacitance of the connector output, C_{wm1} are C_{wm2} are metal coupling capacitances, C_{d_in} is the diffusion capacitance of input driver and C_{g_out} is the gate capacitance of output driver.

3.2.3. Power Model of the Arbiter

In previous work [26,39], three different arbiters have been modeled: matrix, round-robin and queuing. In SCMP, the matrix arbiter is modified to implement round robin operation [40]. The implementation of the circuit is shown in Figure 3.5. Consider a

matrix arbiter with N request inputs. A binary matrix is constructed by setting the element in i^{th} row and j^{th} column to 1 if the input i has higher priority than that input j . Since, the matrix is symmetric, only $N(N-1)/2$ flip-flops are needed to store the binary values specifying the priorities. In a round-robin arbiter, N sets of priority matrices are present and the chosen matrix is rotated using a one-hot pointer. The matrix priorities are fixed. So, flip-flops are not required.

The power dissipation in a matrix arbiter with N request inputs, with no flip-flop storage elements is derived from the equations in [39].

$$P_{arbiter} = f_{clk} \times (P_f/L_p) \times E_{arbiter} \quad (3.20)$$

$$E_{arbiter} = (N-1) \times E_{priority} + N \times (N-1) \times E_{int} + E_{req} + E_{gnt} + E_{cb_ctr} \quad (3.21)$$

where, f_{clk} is the clock frequency, P_f is the flit arrival rate, L_p is the average packet length in flits, $E_{priority}$ is the energy dissipated in the priority logic capacitance, E_{int} energy dissipated in the internal capacitance, E_{req} is the energy dissipated in the request lines, E_{gnt} is the energy dissipated in the grant lines and E_{cb_ctr} is the energy dissipated due to loading of crossbar control signals by grant signals.

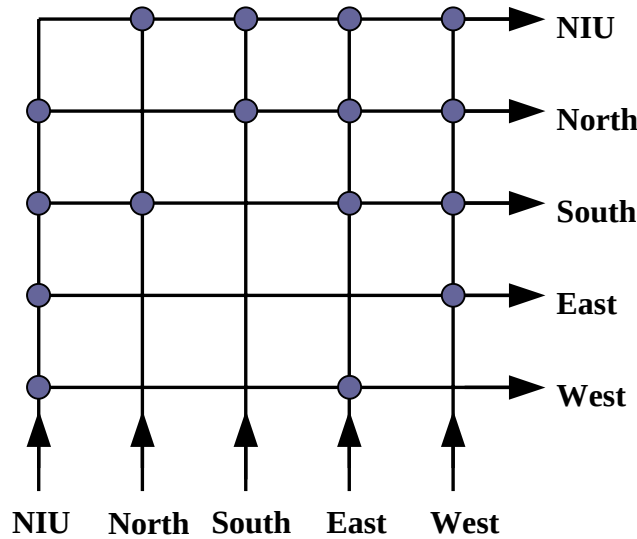


Figure 3.4: A 5×5 Matrix Crossbar Switch

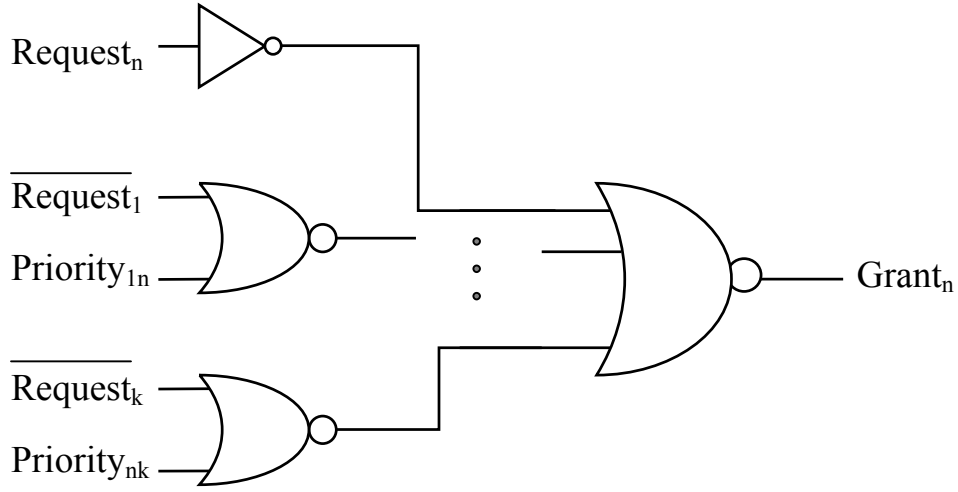


Figure 3.5: Matrix Arbiter Modified for Round Robin Operation

3.3. Scaling Models to DSM Technologies

Single Chip Parallel Computers are designed for future DSM technologies. Hence, it is important that power estimation and analysis of implementations targeted to future scaled technologies be possible. Since, the device parameters and characteristics of these technologies are not available the power models are developed for the available feature sizes and then power models are scaled for technology based on projections. In dynamic power dissipation, capacitance is the factor that changes with device technology. Hence, the trends in capacitance are the most important to consider, while scaling the models to DSM technologies. Other factors like supply voltage, frequency, resistance etc also need to be scaled. The projections in the ITRS [13] are considered for most of the factors.

Considerable research work is focused on studying the trends in capacitance [12,42]. Capacitance estimation is not only important for power analysis, it is also important for delay analysis. In [12], it was studied that trends shown in 1997 SIA Roadmap [41] are misleading and more pessimistic as far as global wire scaling is concerned. In [12], aggressive and conservative predictions were made, instead of just a single prediction.

The wire metrics are predicted under three different categories: local, semi-global and global. In this work the median value between aggressive and conservative predictions was used to obtain different technology scaling factors for local, semi-global and global categories. The data for wire dimensions scaling, resistance scaling and capacitance scaling from [12] are presented in Tables 3.2, 3.3 and 3.4, respectively.

L_{drawn}	0.18 μm	0.13 μm	0.10 μm	0.07 μm	0.05 μm	0.035 μm
Semi-global pitch, μm	0.36	0.26	0.20	0.14	0.10	0.07
Global Pitch, μm	0.72	0.52	0.40	0.28	0.20	0.14
Chip Edge, mm	19	20.7	22.8	24.9	27.4	30.1

Table 3.2: Wire Dimensions Scaling [12]

		0.18 μm	0.13 μm	0.10 μm	0.07 μm	0.05 μm	0.035 μm
conservative	Global Aspect Ratio	2.2	2.3	2.5	2.5	2.5	2.5
conservative	Semi-global ($\Omega/\mu\text{m}$)	96	184	307	627	1220	2509
conservative	Global ($\Omega/\mu\text{m}$)	20	37	58	118	231	473
aggressive	Global Aspect Ratio	2.2	2.5	2.7	2.8	3	3
aggressive	Semi-global ($\Omega/\mu\text{m}$)	96	168	260	340	600	1224
aggressive	Global ($\Omega/\mu\text{m}$)	20	35	54	82	150	306

Table 3.3: Resistance Scaling [12]

		0.18 μm	0.13 μm	0.10 μm	0.07 μm	0.05 μm	0.035 μm
conservative	Sideways ϵ_r	3.75	3.375	3.038	2.734	2.460	2.214
conservative	Semi-global (fF/ μm)	414	387	359	333	311	295
conservative	Global (fF/ μm)	440	423	413	381	355	334
aggressive	Sideways ϵ_r	3.75	2.5	2	1.75	1.5	1.4
aggressive	Semi-global (fF/ μm)	414	343	314	307	296	287
aggressive	Global (fF/ μm)	440	370	335	312	296	287

Table 3.4: Capacitance Scaling [12]

3.4. Leakage Power Modeling

3.4.1. Leakage Modeling for a Single Transistor

The leakage models are developed based on the leakage simulator, *HotLeakage* [25]. The leakage models are constructed from the BSIM3v3.2 [32] equation for leakage in a MOSFET transistor. The leakage model for a single transistor can be given by the following equation [25].

$$I_{leakage} = \mu_0 \cdot C_{ox} \cdot (W/L) \cdot e^{a+b(V_{dd} - V_{dd0})} \cdot V_t^2 \cdot (1 - e^{-V_{dd}/V_t}) \cdot e^{((-|V_{th}| + c(T-T_0) - V_{off})/(n \cdot V_t))} \quad (3.22)$$

where, μ_0 is the zero bias mobility, $C_{ox} = \epsilon_0/t_{ox}$ is gate oxide capacitance per unit area, W/L is the aspect ratio of the transistor, e^a is the adjust parameter to fit the simulation result, $e^{b(V_{dd}-V_{dd0})}$ is the DIBL factor, V_{dd} is the specified supply voltage, V_{dd0} is the default supply voltage for each technology, $V_t = kT/q$ is the thermal voltage, V_{th} is the specified transistor threshold voltage, V_{th0} is zero bias threshold voltage in the room temperature $T_0 = 300K$, T is the specified temperature, c is the temperature factor of threshold voltage, n is the sub-threshold swing coefficient, V_{off} is an empirically parameter.

Before the start of a simulation, the user provides some parameters like μ_0 , t_{ox} , V_{dd} and V_{th} and W/L . The user also specifies the technology node for the simulation. For the specified device technology, several transistor level parameters (a , b , c , V_{off} , n) should be determined by the curve-fitting methods. Transistor level simulation is needed for both NMOS and PMOS for determining the required curve-fitting parameters. Transistor level simulations can be performed using Spice or Spectre simulator. Several iterations are required for good curve fitting. The curve fitting parameters derived in [25] were used in this work.

3.4.2. Leakage Modeling for a Design Cell

The Leakage model for a design cell is constructed using a methodology similar to that followed for a transistor leakage model. This approach is similar to that followed in [25]. For a cell, the leakage current can be estimated by the following equation

$$I_{cell} = (n_N \cdot I_N + n_P \cdot I_P) \cdot k_{design} \quad (3.23)$$

where, I_N and I_P are the unit leakage currents calculated based on equation (3.22) when the aspect ratio, W/L , is equal to 1, n_N is the number of NMOS transistors, n_P is the number of PMOS transistors, k_{design} is the design factor determined by the stack effect and aspect ratio of transistors in the cell. To determine the k_{design} factor, transistor level simulation is required for the specific cell. All possible input combinations are considered for the cell in order to determine the design factor.

The model constructed using a single design factor is suitable only when the NMOS and PMOS transistor parameters are close [31]. This is usually not the case and two different design factors, k_n and k_p , should be derived for the N and P transistors. Then the leakage current in a cell will be given by the following equation [25].

$$I_{cell} = n_N \cdot I_N \cdot k_n + n_P \cdot I_P \cdot k_p \quad (3.24)$$

The design factors k_n and k_p must be derived from transistor level simulations. The design factors k_n and k_p are given by the following equations

$$k_n = (I_{1n} + I_{2n} + I_{kn} + \dots) / (N \cdot n_N \cdot I_N) \quad (3.25)$$

$$k_p = (I_{1p} + I_{2p} + I_{kp} + \dots) / (N \cdot n_P \cdot I_P) \quad (3.26)$$

where, N is the number of possible input combinations, $(I_{1n}, I_{2n}, \dots, I_{kn}, \dots)$ are the leakage currents when the pull-down network is turned off and $(I_{1p}, I_{2p}, \dots, I_{kp}, \dots)$ are

the leakage currents when the pull-up network is turned off. For a given cell, all possible inputs can be divided into two groups: one group of inputs that will turn off the pull-down network which composed of NMOS transistors; another group of inputs that will turn off the pull-up network which is composed of PMOS transistors. So, the leakage currents measured from simulation can be divided into two groups and the design factors can be estimated.

Using this methodology leakage models are constructed for the basic cells and then leakage power in an architectural module is determined from leakage power dissipated in the basic cells. Constructing the leakage model for a cell is a time consuming and cumbersome process. Hence, leakage models are constructed only for major components in the SCMP system like array structures. It is easy to construct the leakage model for array structures because of the repetitive nature of their implementation. If the leakage power model for the leaf cell in a module is constructed, it is easy to estimate the leakage power dissipated in that module.

As seen in Section 2.1.3, it is important to model Gate Leakage because it is increasing significantly with technology scaling due to direct tunneling current. Gate Leakage estimation models have been adapted from Gate Leakage models based on BSIM4 in HotLeakage. Another important aspect to consider is parameter variations. Based on the approach followed in [25] inter-die variations are taken into account in the architectural simulations. In inter-die variation it is assumed that variations affect all devices on a chip equally. In intra-die variations mismatch in behavior of devices in the same chip is assumed but it is difficult to model intra-die variations. Hence, only inter-die variations are modeled.

A single variation component with a mean and a variance is sufficient to model inter-die variations. Variations are taken into account for the length of the transistor, thickness of the gate oxide, supply voltage and threshold voltage of the transistor. For each of the four parameters, the user can specify the mean μ , variance σ and the number of samples N . During the initialization of the simulator, the N gaussian distribution samples are generated and the corresponding leakage currents are calculated. The mean of the leakage currents is used in simulations to show the affect of parameter variations.

Chapter 4

4. The Power Analysis Tool

Power optimization at the architectural level requires an infrastructure that can analyze power and performance ramifications of different architectural choices. A simulator that can quantify both power and performance with reasonable accuracy is required to study the design trade-offs and make the appropriate design choices. The microarchitectural simulator modeling the SCMP system is presented in this chapter. The integration of the dynamic and static power models of the SCMP architecture into the microarchitectural power and performance simulator is explained. The metering options available in the simulator for efficient power-performance analysis are discussed.

4.1. The Performance Simulator

A cycle-accurate detailed microarchitectural model of the SCMP system enables extensive performance analysis and also aids in analyzing the power-performance trade-offs involved in the design of the SCMP system. The performance simulator is written in C and is made as flexible as possible in order to easily accommodate changes. The simulator models the processor nodes and also the network. For any simulation run the user can configure the SCMP system. Specifications like the number of nodes, the X and Y distribution of nodes, the size of memory, the number of contexts per node, and the instruction cache parameters can be specified through a configuration file.

4.1.1. Processor Core Simulation

The processing core contains the Pipeline, the Register Contexts, the CMT, the CMT Control Logic, the Memory, the Memory Controller and the Instruction Cache. Though the NIU is a part of the processor core, it is considered under network simulation. The

performance simulator contains one or more modules for each of these processor subsystems. The detailed microarchitectural functionality of the processor subsystems is modeled in their respective software modules. Short descriptions of the important modules are provided to enable better understanding of the construction of the performance simulator.

- `node`: The `node` module defines the node structure used by all other modules. An instance of `node` contains instances for the Instruction Cache, the Register Contexts, the CMT array, the Memory, the Memory Controller, the Pipeline, the NIU, the Router and other components for the node it models. Every entry routine into another module requires a pointer to the `node` it will act on. The user may specify the number of nodes along the X and Y axis independently through the configuration file.
- `sim`: The `sim` module simulates the flow of instructions through the SCMP pipeline. In the detailed micro-architectural simulator, `sim` models each pipeline stage as instructions move through them. The functionality of each pipeline stage is modeled as a C routine. For every clock cycle, the routines are called in an order opposite to that of data flow. Executing the pipeline routines in reverse order maintains correct interstage data synchronization.
- `execute`: The `execute` module provides a C routine for every functional unit operation.
- `oscall`: To model low-level I/O primitives such as *fputc*, and other file operations, the simulator uses the `oscall` module to provide an interface to the I/O primitives of the host system on which the simulator is running. This module allows applications running on the performance simulator access to all the low-level standard C I/O functions.
- `dispatch`: Before the `sim` module can access the `execute` or `oscall` modules, the instruction's arguments must be extracted, converted to C variables and passed in as arguments to the appropriate routine. After the execution of an instruction has been simulated, the results must be converted back to the proper format and stored. The `dispatch` module provides a layer between the `sim` module and the `execute` and `oscall` modules to perform such conversions.

- `memory`: The `memory` module models the on-chip local memory. It interprets memory-mapped addresses, and arbitrates memory access among the instruction cache, pipeline and NIU. The user may specify the size of the memory on each node through the configuration file.
- `context`: The `context` module models a processor node's Register Contexts. There are 32 registers per integer context, and 16 registers per floating-point context. However, the user may specify the number of contexts per node through the configuration file. The same set of registers in a context can be used as 32 integer registers or 16 floating point registers.
- `cmt`: The `cmt` module models the CMT array and the CMT control logic. It maintains a table modeling the CMT array and contains routines for updating the table and providing data from the table for use by the modules modeling the Pipeline, the Memory Controller and the NIU.

4.1.2. Network Simulation

One of the most important requirements in a performance simulator modeling a Single Chip Parallel Computer is the inclusion of network simulation. Many network simulators do not model the processors that generate the network traffic. Similarly, many simulators modeling message-passing systems omit the details involved in the transmission of messages from one processor to another. The SCMP microarchitectural simulator, however, models the details of the network and the processing units simultaneously. Therefore, the performance simulator is able to account for the effects of processor loads on the network and the effects of network congestion on the processors.

The SCMP networking hardware includes the NIU and the router subsystems on each node. These subsystems move flits through the network by copying them from one hardware buffer to another. To ensure an accurate simulation of network performance, the simulator modules for these subsystems, `niu` and `router`, replicate each hardware buffer with a software buffer. The `niu` and `router` modules reproduce each flit movement in hardware as a copy from one software buffer to another. By accurately

modeling flit movements, the simulator accounts for the effects of network traffic and gridlock better than a statistical model of the network.

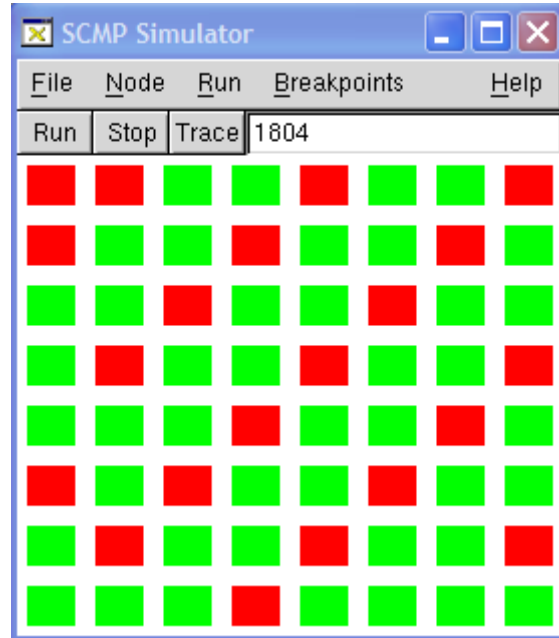


Figure 4.1: SCMP GUI Main View

4.1.3. Graphical Interface

The SCMP microarchitectural performance simulator provides a Graphical User Interface, or GUI, that allows the user to dynamically view the execution of programs in the processor simulator. Each SCMP node is represented by a colored square in the main view of the simulator GUI, shown in Figure 4.1. When the pipeline of a node stalls, the GUI changes the node's square from green to red. When a node has no threads to execute, the GUI draws the node's square in black. Commands are available to run user programs, to trace programs one instruction at a time, and set breakpoints. Users may see more information about a node by clicking its square. This opens the detailed node view shown in Figure 4.2. The detailed node window has panes displaying the contents of the

node's special registers, status, pipeline registers, CMT, and active integer and floating-point contexts. The CMT pane displays information for the active thread context in green, inactive thread or floating-point context in red, and unused contexts in black. All other panes refer to the active thread context and its floating-point context if applicable. Users may also open other windows to examine the memory on a node or other thread contexts.

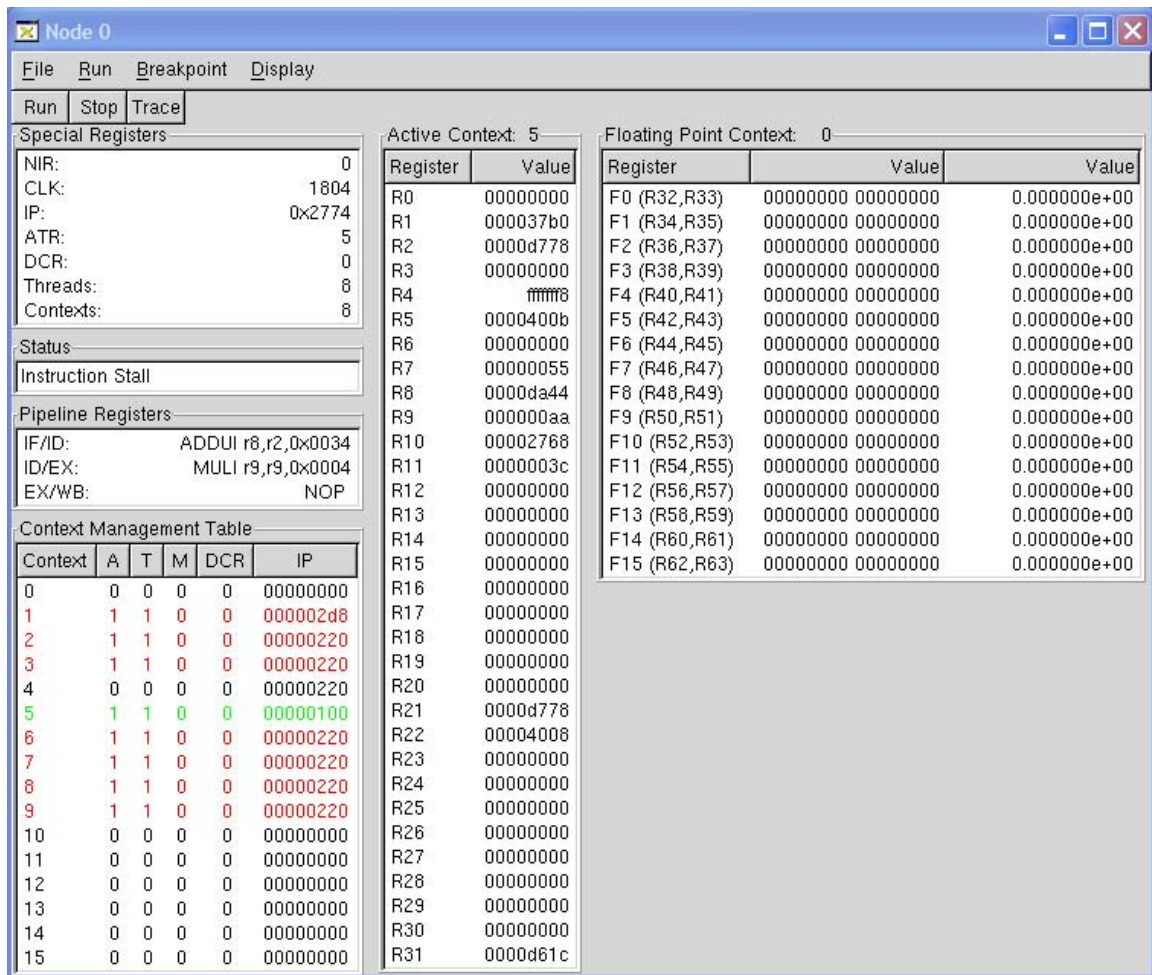


Figure 4.2: SCMP GUI Node Detail View, displaying node '0'

4.2. The Power and Performance Simulator

High-level power estimation is made possible by integrating parameterizable power models into the microarchitectural performance simulator. Dynamic power dissipation in CMOS circuits depends on the signal switching, while static power dissipation depends on the state of the circuit rather than signal switching. So, based on the execution of operations in every clock cycle, the switching activity and hence dynamic power dissipation is estimated. Only the capacitive switching power dissipation component of dynamic power dissipation is considered. As discussed in Section 2.1.1, the capacitive switching power and hence the dynamic power dissipation is given by the equation (2.3). From the equation, the variables required in the calculation of dynamic power are the supply voltage, operating frequency, switching capacitance and the activity factor. The supply voltage and frequency can be specified at the start of simulation. The most important part of power estimation is estimating the switching capacitance and the activity factor.

4.2.1. Construction of the Power and Performance Simulator

The switching capacitance is calculated for different hardware subsystems as explained in Section 3.1 and Section 3.2. The capacitance estimation is performed as the initialization process of simulation. In order to run a power and performance simulation the user has to specify important system configuration parameters in the configuration file. Parameters like number of processor nodes, memory size, number of contexts per node etc are provided along with technology node and frequency of operation. The supply voltage is determined based on the technology selected. The frequency of operation should also be verified, to determine if it is greater than the maximum possible operating frequency.

In order to estimate the activity factor, access counters are added to keep track of the number of accesses, per cycle, to each hardware structure. The micro-architectural simulator has a modular structure. Each hardware subsystem is modeled with a software module, which uses software routines to represent the internal operations of the

subsystem. The modular structure of the simulator enables easy inclusion of access counters. Updates incrementing the access counter of a particular hardware structure are included in the microarchitectural performance simulator code in all the places where the hardware is accessed. Along with the activity factors bit-line population counts of the data being read/written to array structures or system buses are computed. The population counts keep track of the bit switching or signal switching in a bus or read/writes to array structures. Population counts are not calculated for all components, as they result in a large overhead and degrade simulator speed. Hence, they are computed only for components where they significantly influence the power dissipation. Access counters are included and updated for all components.

The power dissipation in the network subsystems is also estimated using the approach similar to that used for the processor subsystems. Switching capacitances are estimated as a part of initialization and then activity factors are estimated using access counters and population counts. The power analysis tool also estimates leakage power dissipation. Based on the provided configuration parameters at the start of simulation leakage current is obtained from the leakage models and is averaged for parameter variations. The average leakage power dissipation is calculated as a part of the initialization process. The total leakage power is added to the total dynamic power to obtain the power dissipated in the SCMP system.

4.2.2. Power and Performance Simulation

The power and performance simulator can be used to obtain power and performance numbers corresponding to a specific system configuration and a specific benchmark. The parameters required for specifying the system configuration and those required for initializing the dynamic and static power estimation models have to be provided in the configuration file. The configuration parameters are read from the configuration file during simulator initialization. The switching capacitance for the dynamic power and the leakage currents for static power are estimated as a part of the initialization process. After initialization, the execution of the benchmark starts in the simulator. The microarchitectural simulator is a cycle accurate simulator. As instructions get executed and operations are performed in the hardware modules, the access counters keep track of

the nature and number of accesses to the hardware subsystems in a given clock cycle. At the end of the simulation for the clock-cycle, the access counters and populations counts are used to estimate the dynamic power dissipated in that clock cycle. The average power dissipated and the maximum cycle power dissipated in each hardware subsystem in each processor node is updated every clock cycle. The average and maximum power dissipated in each processor node, in the network and in the entire system is updated. The static power dissipated is also taken into account to estimate the total power dissipation.

4.3. Power Analysis Tool Options

Extensive power and performance analysis requires a simulator that is flexible, configurable and offers a wide variety of power and performance metering options. Due to its modular construction, the microarchitectural simulator can be modified with ease and flexibility. The simulator provides the ability to configure the SCMP system and its implementation for every simulation run. This allows the designer to study the variation of power and performance for different architectural level design choices. The power and performance simulator also offers a number of power and performance metering options that can be utilized for analyzing the different design aspects.

On execution of a workload or benchmark, the performance simulator reports the number of instructions and the total number of clock-cycles for the execution a given benchmark. These numbers can be used to compute the performance in MIPS, MOPS, Clock cycles per Instruction (CPI) or Instructions per Clock cycle (IPC). The simulator can also generate a number of log files to measure the performance of the system. Some of the files contain summary information and are only written at the end of the simulation. Others show the changes in system performance over time. For these files, an entry is written after a given time interval which can be specified in the configuration file. The log file is written if the user selects the desired metering or log file in the configuration file, before the simulation starts.

The power simulator also offers a number of metering options. The maximum power dissipation and average power dissipation of the system for a specified configuration and

implementation of the system can be estimated for different benchmarks. The variation of power dissipation with important system parameters can be studied. Figure 4.3 shows the variation of average power dissipation of the SCMP system with number of processor nodes, for three benchmarks. As power and performance numbers are both available the energy efficiency or MIPS/Watt can also be explored for an energy efficient configuration. Energy efficiency is crucial in battery-operated systems.

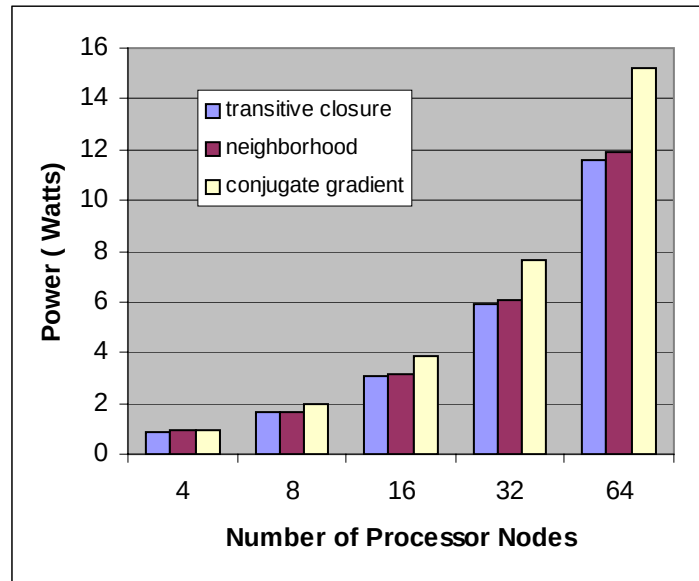


Figure 4.3: Average Power Dissipation for Three Benchmarks

The maximum power and average power dissipated in each processor node and in each subsystem within that node can be accessed. The maximum power dissipated by the processor nodes can be written to a Matlab M-file. The M-file can be executed in the Matlab environment to obtain plots that can be analyzed to locate relatively hotter areas on the chip, as shown in Figure 4.4. The distribution of power within a processor node can be analyzed to isolate subsystems that are major contributors to power dissipation and focus optimization efforts on them. Figure 4.5 shows an example.

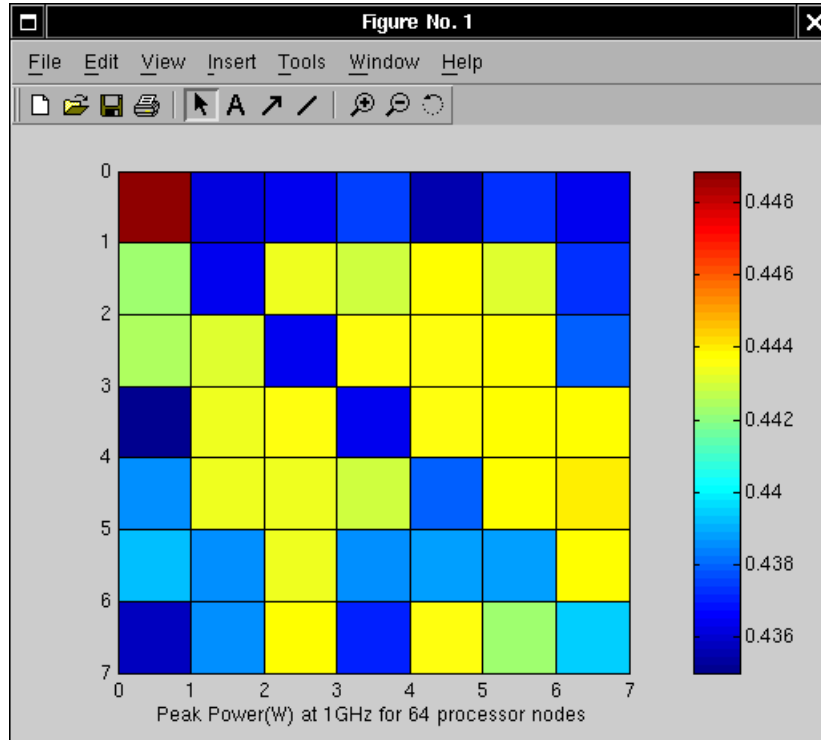


Figure 4.4: Peak Power Dissipated for the 64-node configuration at 1 GHz

The power simulator also offers different conditional clocking options, similar to that in [22], which enables us to analyze the effects of clock gating. Three different conditional clocking options are available.

- i) No Conditional Clocking: The hardware structure is assumed to consume full power irrespective of whether it is accessed or not accessed during that clock cycle.
- ii) Aggressive Conditional Clocking: The hardware structure is assumed to consume full power if it is accessed and zero power if it is not accessed during that clock cycle.
- iii) Conservative Conditional Clocking: The hardware structure is assumed to consume full power if it is accessed or 10% of full power if it is not accessed during that clock cycle.

Conservative conditional clocking is more practical if the effect of clock gating has to be studied because it is not possible to fully turn off a hardware component. The power simulator also provides the option of a power metering file that provides the maximum power dissipated and the average power dissipated in a metering interval that can be specified in terms of number of clock cycles in the configuration file. The output of the power meter is shown in Figure 4.6 and the corresponding graphical representation of the power metering data is presented in Figure 4.7. So, we can see that the power-performance simulator enables extensive analysis of power, performance and energy efficiency for high-level hardware and software optimizations.

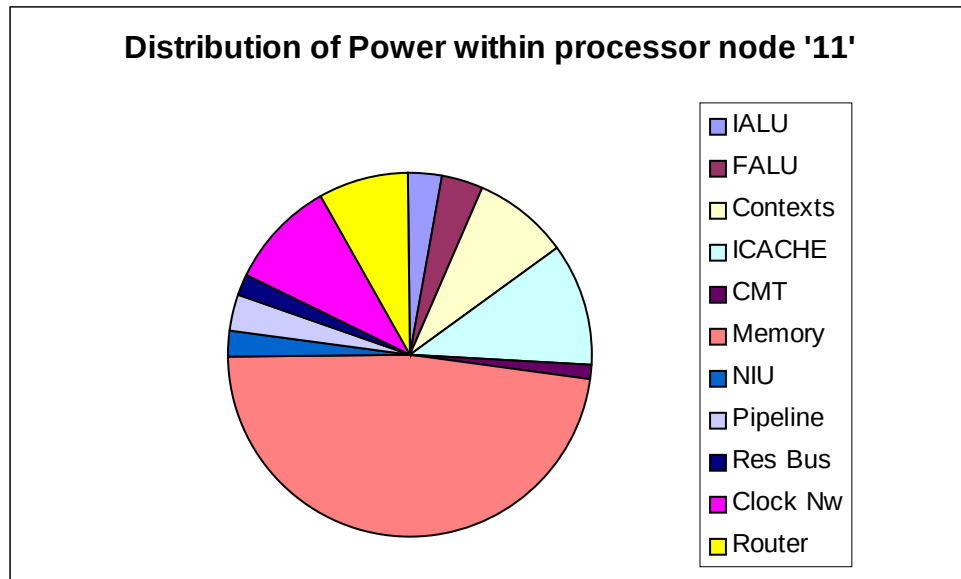


Figure 4.5: Distribution of Power Among Subsystems for a Single Node.
(Results for Transitive Closure Benchmark)


```
=====
# Power Metering:
# Average and Maximum power (in Watts) dissipated by the processor

# Average Power = Sum_total/Meter_Interval
# Format:
# Clock          Sum_total          Average          Maximum
# -----
#
1000000          2.33625e+06          2.34W           6.42124W
2000000          2.29473e+06          2.29W           2.82272W
3000000          2.29428e+06          2.29W           2.82198W
4000000          2.29412e+06          2.29W           2.82173W
5000000          2.29405e+06          2.29W           2.82161W
6000000          2.29398e+06          2.29W           2.82153W
7000000          3.02029e+06          3.02W           5.7163W
8000000          3.32734e+06          3.33W           5.92922W
8980551          2.42319e+06          2.47W           4.26222W
=====
```

Figure 4.6: Example of the Power Metering File

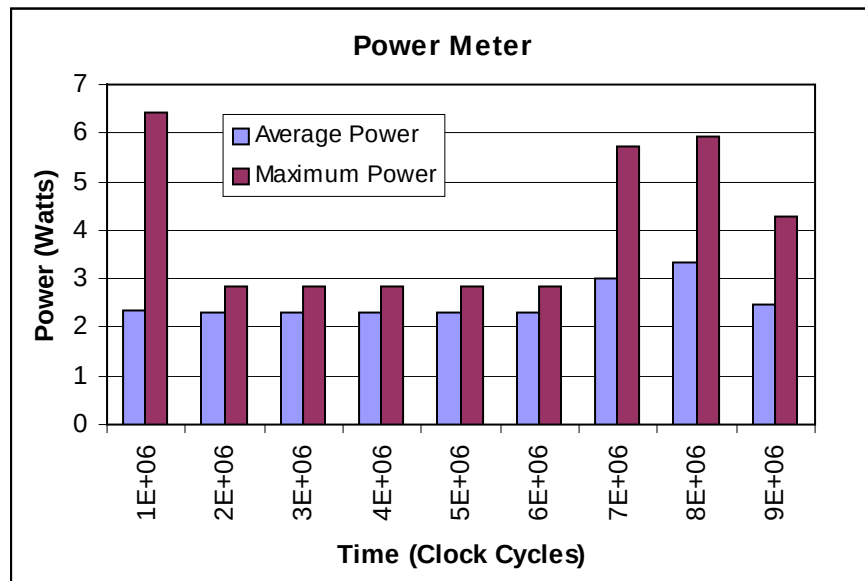


Figure 4.7: Graphical Representation of the Power Metering Data

Chapter 5

5. Power Analysis & Optimizations

The power and performance analysis tool and the power analysis methodology developed in the previous chapters are used to analyze the SCMP architecture. The power advantages inherent in the SCMP architecture are explored in this chapter. Some power optimizations are proposed. The simulation results for energy-efficiency, power and performance of the SCMP architecture are presented in this chapter. The design trade-offs involved in the architectural level and system level design of the SCMP processor in specific and single chip parallel computers in general, are discussed.

5.1. Power Advantages in the SCMP Architecture

5.1.1. Thread Level Parallelism

SCMP exploits Thread Level Parallelism (TLP) to achieve a high throughput. It does not rely on aggressive speculation or out-of-order execution. Each processor in SCMP is a simple RISC processor with in-order execution. While higher performance processors could be used, there are power advantages to using in-order execution. In fact, in tests that were designed to make a processor consume maximum power, the distribution of power dissipation among the components in a PentiumPro processor showed that the components necessary to perform dynamic scheduling and out-of-order execution contributed to a major portion of the total power [14].

In [14] a particular form of speculation control is used for reducing unnecessary work. Results show that by limiting speculation, a significant reduction in energy consumption can be achieved. SCMP does not depend on any form of speculation; hence resulting in large energy savings. It has been studied that the overhead and energy costs

in superscalar machines are worse than that in simple RISC machines, and overwhelm the performance gains obtained through Instruction Level Parallelism (ILP) [15]. So, a system that consists of simple processors on a chip executing multiple threads in parallel can achieve high levels of performance at low energy costs.

5.1.2. Absence of Global Signals

In SCMP there are no global signals or shared resources. Each processor node has its own clock. Hence, the clock is globally asynchronous and locally synchronous. Typically the clock is the largest power-consuming component in a high performance processor [43]. This includes the clock generator, the clock drivers and the clock distribution tree. The effects of Globally Asynchronous Locally Synchronous architectures on clock power consumption were evaluated in [44]. Power savings of up to 70% in the clock-net, corresponding to 20% in overall dissipation compared to conventional globally synchronous designs, were calculated.

Having a globally asynchronous and locally synchronous clock in SCMP is an advantage that is not readily available to traditional superscalar processors. The clock network within each processor is also small enough that it does not consume much power. In deep submicron technologies, global signals cannot support very high clock speeds due to increased wire delays. One approach to decrease delays is to use repeaters. However, use of repeaters can lead to huge power penalties [11]. The absence of global signals in SCMP enables the system to support high clock frequencies without using repeaters. Hence, SCMP provides a power efficient, high performance solution for future technologies.

5.1.3. Uniform Power Density

In most high performance processors, one factor that exacerbates the thermal management problem is that local areas of the die, depending on where different functions are executed, have much higher power densities than the average power density [45, 46]. These power density variations lead to a higher heat-flux concentration in certain areas of the die and lower heat fluxes in other regions on the die, causing large

temperature gradients on the die. This issue is becoming increasingly important with the emerging generation of high complexity and high integration density processor systems. The thermal designs have to meet stringent heat-flux constraints that are significantly higher than the average heat flux at the silicon-package interface [45].

The architectural plan of having multiple processors, all consuming considerably less power compared to any of the high-performance processor cores, provides a low on-die power density in SCMP. In SCMP, the processors are simple in design and consume less amount of power. For most applications the workload is evenly distributed among the processors. At any given time, the processor nodes may have different instantaneous power dissipation but their average power dissipation is almost equal. The total average power dissipation and the maximum power dissipation of the chip is the added sum of the power dissipated by all the processors and that dissipated in the network. Hence, there are no local areas of very high or very low power densities than the average power density.

5.1.4. Power Performance Scaling

In most high performance systems, a huge power penalty has to be paid for even a moderate increase in performance. Power increases not linearly but almost exponentially with performance. In SCMP performance can be improved by increasing the number of processor nodes. The processor nodes are identical, so with a uniform workload distribution, their average power dissipation should be the same. Simulation results show that in SCMP power dissipation will not scale as drastically with performance. Power increases almost linearly as opposed exponentially with performance. For some benchmarks, power efficiency increases with performance.

5.2. Power Optimizations

The modular architecture of SCMP enables easy implementation of various architecture level power optimizations. These power optimizations result in significant power savings with negligible loss in performance.

5.2.1. Split Register File

Split register file architectures can result in a low power design [47]. In SCMP each processor has multiple contexts. The register file can be split into different register banks such that each context has a separate bank. Splitting the register file will not only reduce energy consumption but also improve performance by reducing the register file access time [48]. The additional logic required can be integrated into the previously available context switching logic. The delay introduced by the register splitting is hidden in the context switching latency.

5.2.2. Clock Gating

Clock gating can result in huge dynamic power reductions. Since each SCMP processor node has its own clock, clock gating can be easily implemented at the node level without problems of glitches or skews. When a node has no threads to be executed, the clock to the node is disabled until a new thread is created on that node. However, it must be noted that not all modules in the node can be completely gated. Modules like the NIU and the CMT control logic need a clock. The clock to the DRAM memory refresh logic cannot be gated. The NIU has to be always ready for an incoming thread and the CMT control logic follows a polling scheme to determine if a new thread has arrived at the NIU. If the CMT control logic determines that a new thread has arrived, it stores the necessary thread data in the CMT and enables the clock to the pipeline and other logic modules required for execution of the thread.

Within each node, clock gating can be used to turn off unused modules. The split register file architecture enables easy implementation of clock gating to gate the clock to inactive contexts. The clock is enabled only for the active context register bank. There is no performance loss due to this implementation because the inherent latency is hidden in the context switches. An important criterion while implementing clock gating is the impact on performance. Increasing the granularity of clock gating may result in significant increase in logic and the corresponding loss in performance. Also, with increasing granularity the ratio of power savings to clock gating logic may decrease. Increases in the levels of clock-gating increases the risk of timing hazards, glitches, clock

skew and synchronization failures. Hence, there is trade-off in power, performance and reliability issues when deciding the level, granularity and nature of clock gating.

5.2.3. Gated VDD

In deep submicron technologies, leakage power is going to be prominent and will dominate power dissipation in a processor. Gated VDD [49] has been suggested as a method for leakage power optimization. Power distribution is also a major concern for processor designs in deep submicron technologies. Each processor node is independent in terms of clock. The possibility of each node having isolated power distribution can also be explored. This would enable easy implementation of leakage optimizations like gated VDD.

5.3. Power, Performance and Energy Efficiency

Power dissipated in the SCMP system is estimated using the power performance simulator, for the 70 nm technology node. Both dynamic and static power dissipation are taken into account. A local memory of 8 MB per processor was considered for power and performance estimation. Since, we are using an architectural level power simulator, we do not expect the power values to be completely accurate, but they provide an estimate of the power dissipation that can be expected from the system. The benchmarks used to evaluate SCMP are taken from DARPA Data Intensive Systems (DIS) stressmark suite [50]. The transitive closure, neighborhood and conjugate gradient benchmarks are selected.

Energy-efficiency or MIPS per watt is considered to be a better metric than power especially in battery operated devices [51,52]. The performance is given in Millions of Operations Per Second (MOPS) because the total number of operations for a benchmark depends on the benchmark algorithm and remains constant across all architectures. It is considered to be independent of clock frequency. Simulation results are obtained for power, performance and energy efficiency for the three benchmarks. Performance and power are plotted for different number of processor nodes at 5GHz clock frequency. The

advantage of SCMP over other architectures in the future will be the ability to run at high clock frequencies. Simulation results for 200Mhz are also included in order to compare with other processors and in order to show the scaling of power and performance with frequency.

5.3.1. Power and Performance

On implementing the split register file architecture and clock gating optimizations presented in Section 5.2, power dissipation decreased by almost 20% across all the benchmarks. The resulting loss in performance is negligible. For power estimation, it is assumed that a percentage of dynamic power is consumed even when the clock is gated. The simulation results presented in this section are those obtained after implementing the power optimizations.

Figure 5.1, 5.2 and 5.3 show the performance and power plots at 5GHz and at 200MHz, for the transitive closure, neighborhood and conjugate gradient benchmarks, respectively. As expected, performance improves with an increase in the number of processor nodes, without a huge power penalty. Good performance is observed across all benchmarks at 5GHz. The performance for the neighborhood benchmark decreases with pixel depth due to a higher communication to computation ratio. The SCMP system running at 200 MHz promises significant performance with low power dissipation.

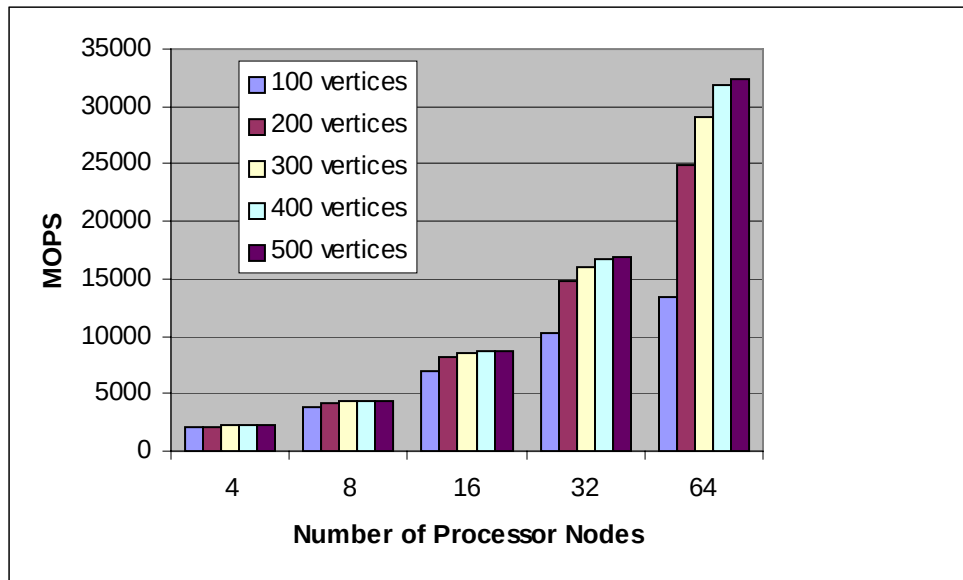


Figure 5.1a: Performance for Transitive Closure at 5GHz

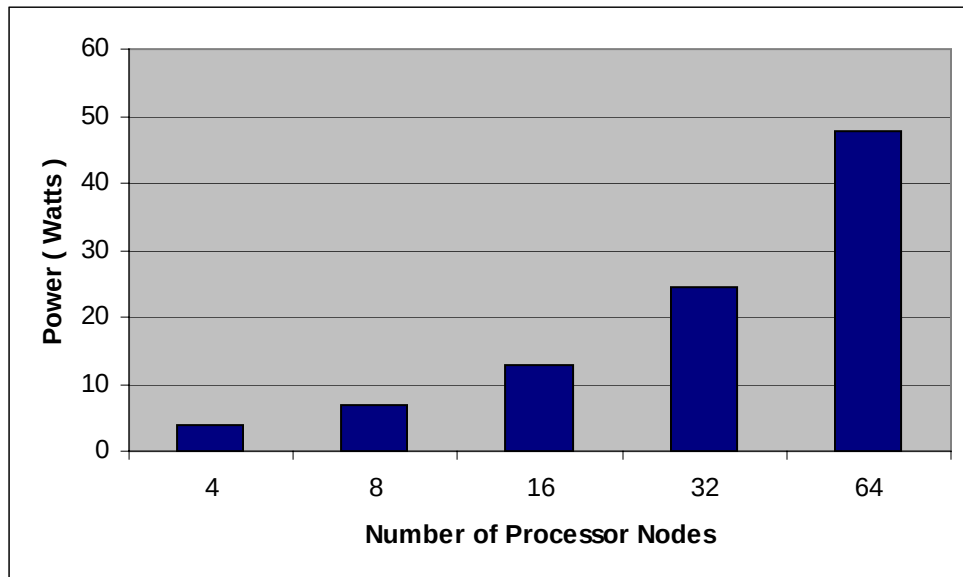


Figure 5.1b: Power for Transitive Closure at 5GHz

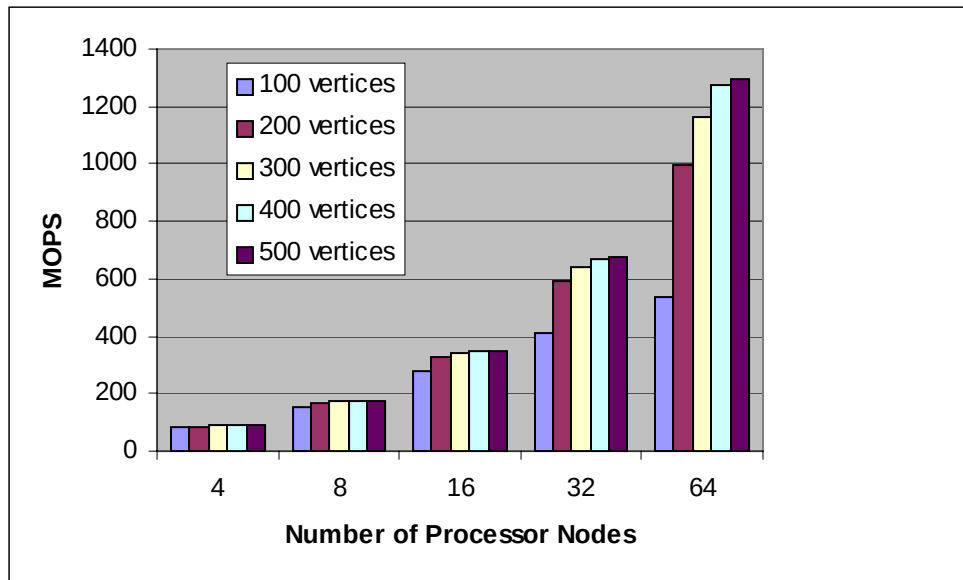


Figure 5.1c: Performance for Transitive Closure at 200MHz

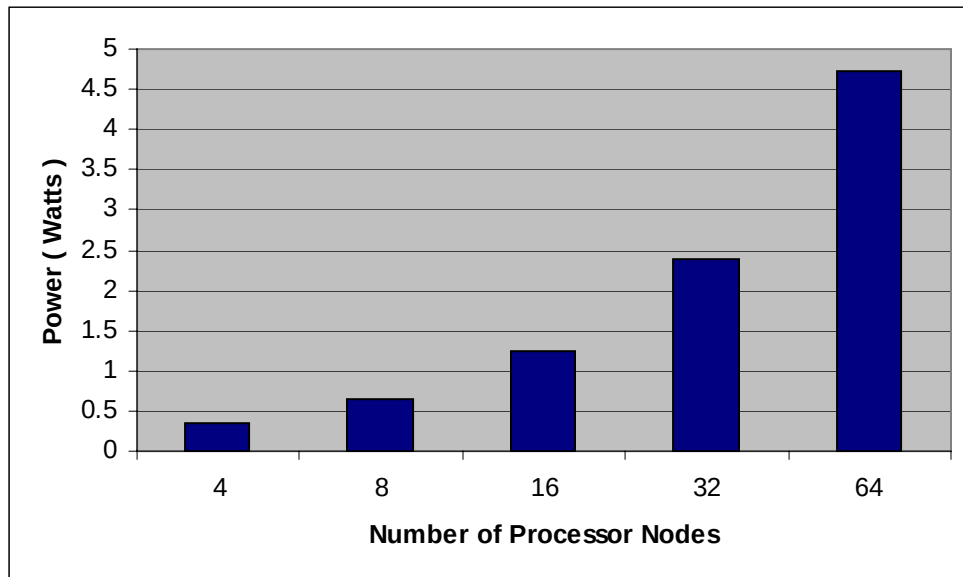


Figure 5.1d: Power for Transitive Closure at 200MHz

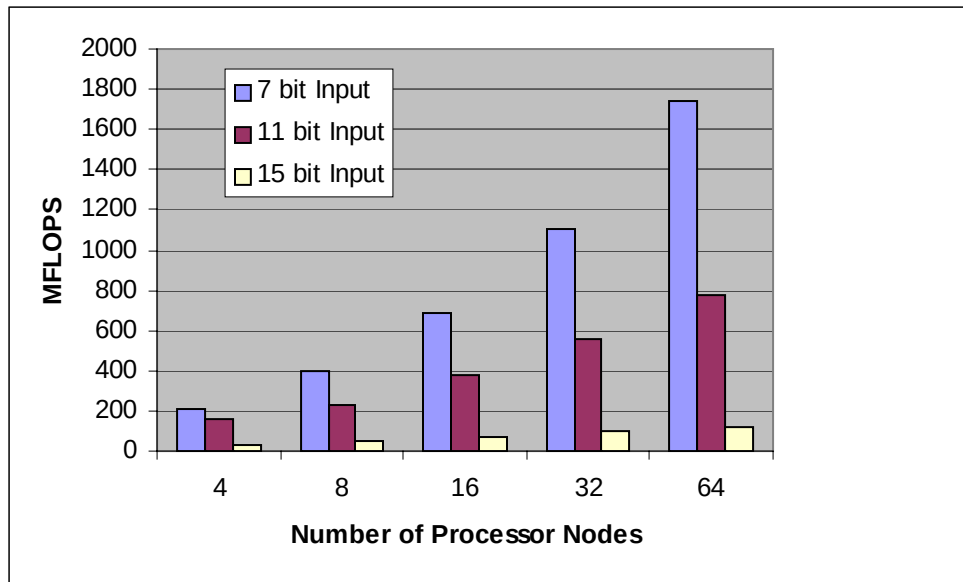


Figure 5.2a: Performance for Neighborhood at 5GHz

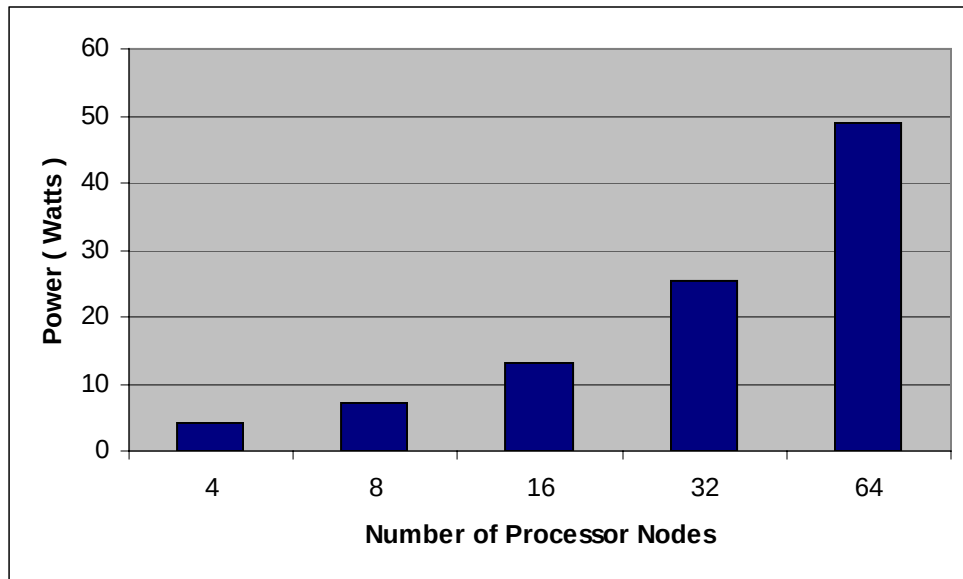


Figure 5.2b: Power for Neighborhood at 5GHz

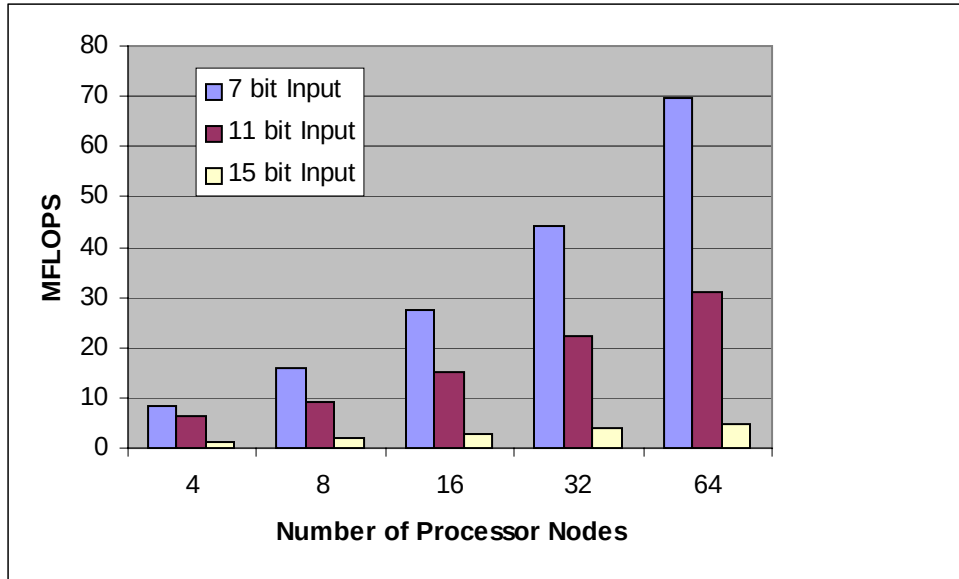


Figure 5.2c: Performance for Neighborhood at 200MHz

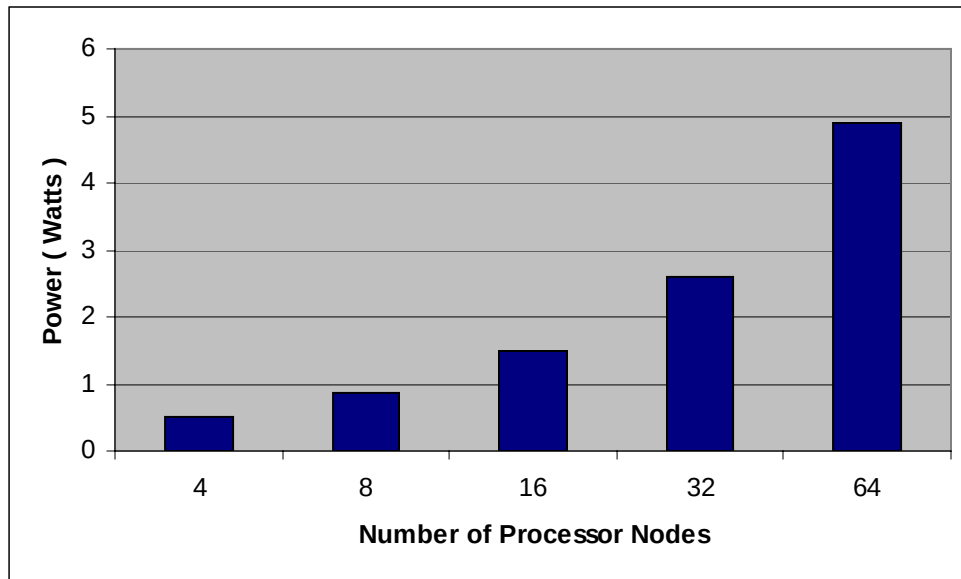


Figure 5.2d: Power for Neighborhood at 200MHz

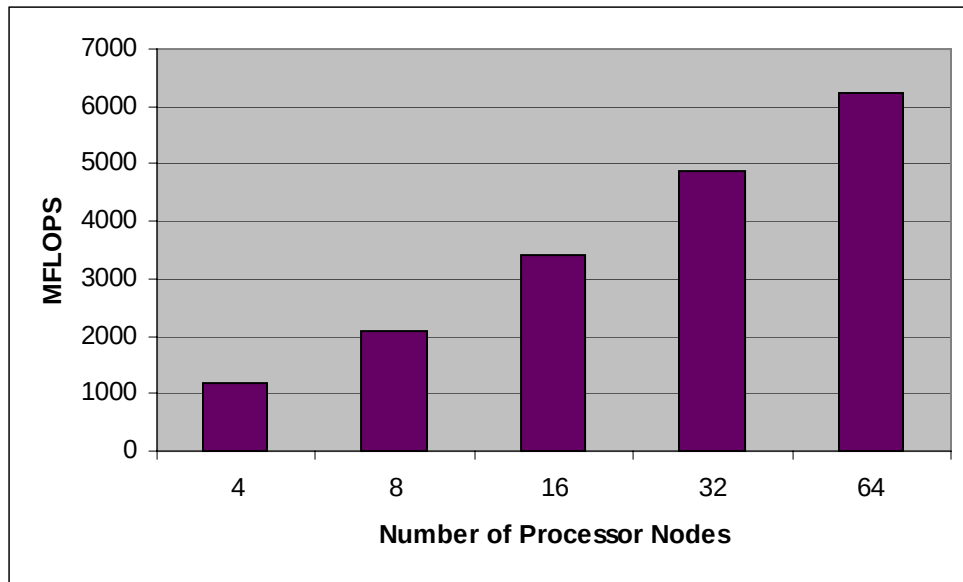


Figure 5.3a: Performance for Conjugate Gradient at 5GHz

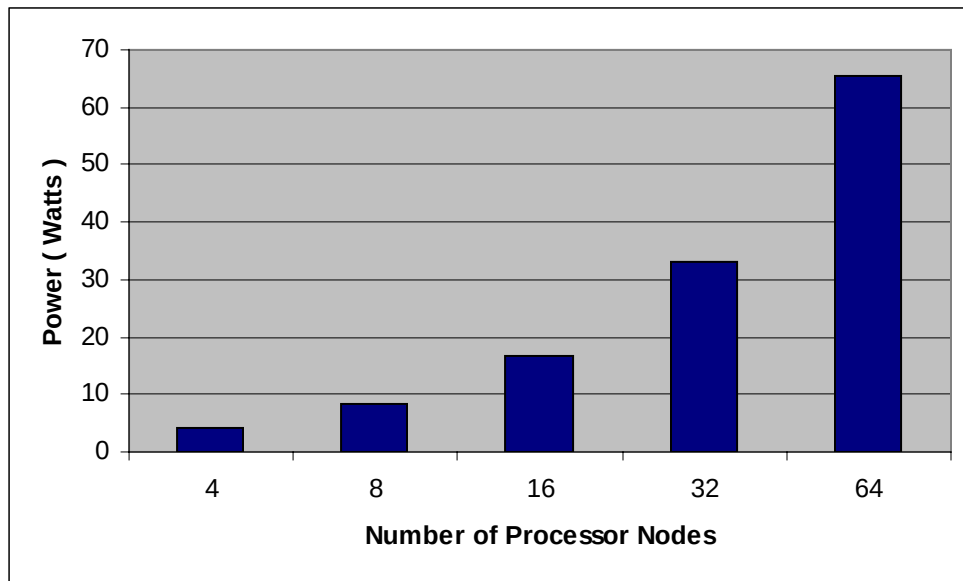


Figure 5.3b: Power for Conjugate Gradient at 5GHz

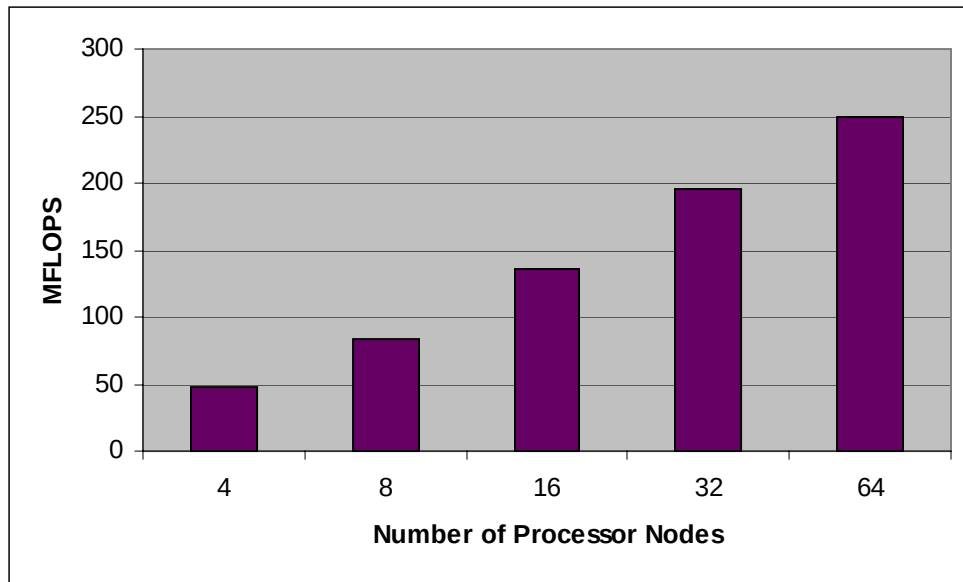


Figure 5.3c: Performance for Conjugate Gradient at 200MHz

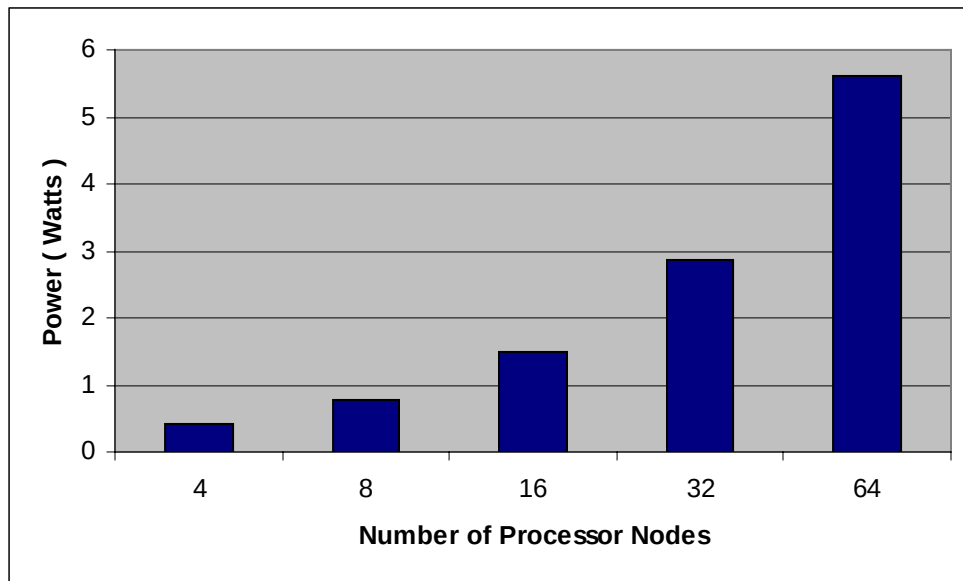


Figure 5.3d: Power for Conjugate Gradient at 200MHz

5.3.2. Energy Efficiency

Energy efficiency is an important metric when battery operated devices are concerned. Better energy efficiency implies that for a given computational task, lesser amount of energy is required from the power source. This leads to longer battery life and lower packaging costs in battery operated mobile devices. Energy efficiency is becoming important beyond mobile processors. Energy efficiency is also important in desktop processors and servers, where consuming less energy from the power supply and conserving energy is becoming crucial [1, 53]. The energy efficiency plots in MOPS/watt for SCMP are shown in Figure 5.4-5.6. SCMP shows good energy efficiency. The variation of energy efficiency with number of processor nodes is application dependent.

It is normally assumed that energy efficiency is independent of frequency and remains constant with variation in frequency [52]. But simulation results show that energy efficiency varies with frequency. As shown in Figure 5.7, energy efficiency decreases at lower frequencies of operation. This happens because energy efficiency is insensitive to frequency only when considering dynamic power dissipation. Energy dissipated in a time period $[0, T]$ is given by the following equation.

$$E = P_{avg} \cdot T \quad (5.1)$$

Where, P_{avg} is the average power dissipated in the time period $[0, T]$. If a clock running at frequency f is considered, the energy dissipated in one clock cycle can be given by the following equation.

$$E = P_{avg} \cdot (1/f) \quad (5.2)$$

Recall from Equation 2.3 that capacitive switching power dissipation is directly proportional to frequency. If P_{avg} is the capacitive switching power, then energy E will be independent of frequency. If P_{avg} is due to a source of power dissipation that is independent of operating frequency, then energy E will increase with decrease in frequency. In this work both static and dynamic power dissipation are considered for

analysis. Static power dissipation drastically increases with technology scaling and is quite significant in the 70nm technology node. Dynamic power dissipation increases with frequency of operation. At very high frequencies, dynamic power dissipation is very high and it dominates the nature of energy efficiency. So, energy efficiency appears to be insensitive to frequency. However, at low operating frequencies, dynamic power dissipation is lesser and static power dissipation becomes a significant percentage of total power dissipation. For a given computational task energy dissipated effectively increases with decrease in frequency and hence energy efficiency degrades (Figure 5.7). Hence, for applications where battery life is a concern, it may be advisable to use higher clock frequencies and an optimal number of nodes that would provide the desired throughput and energy efficiency, while meeting the power constraints.

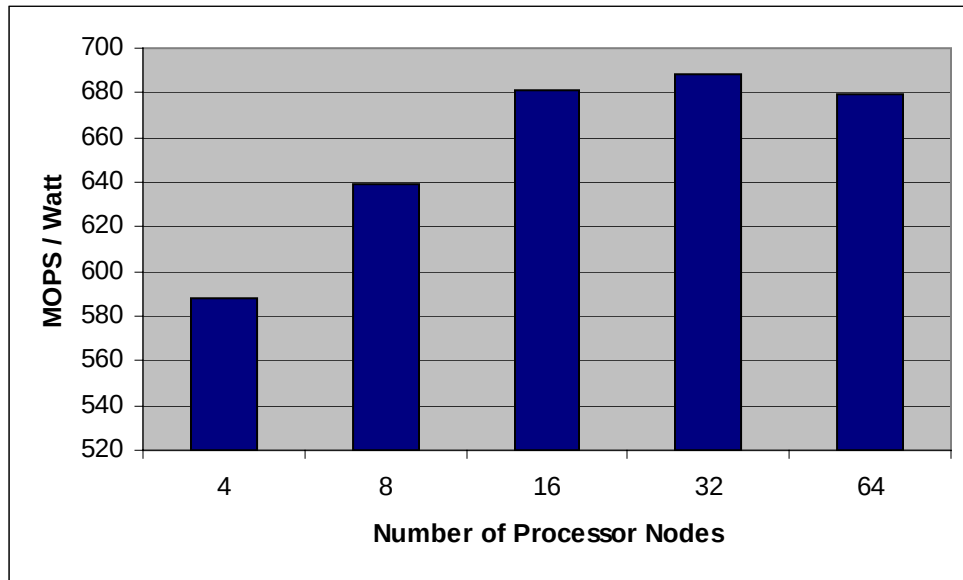


Figure 5.4: Energy Efficiency for Transitive Closure

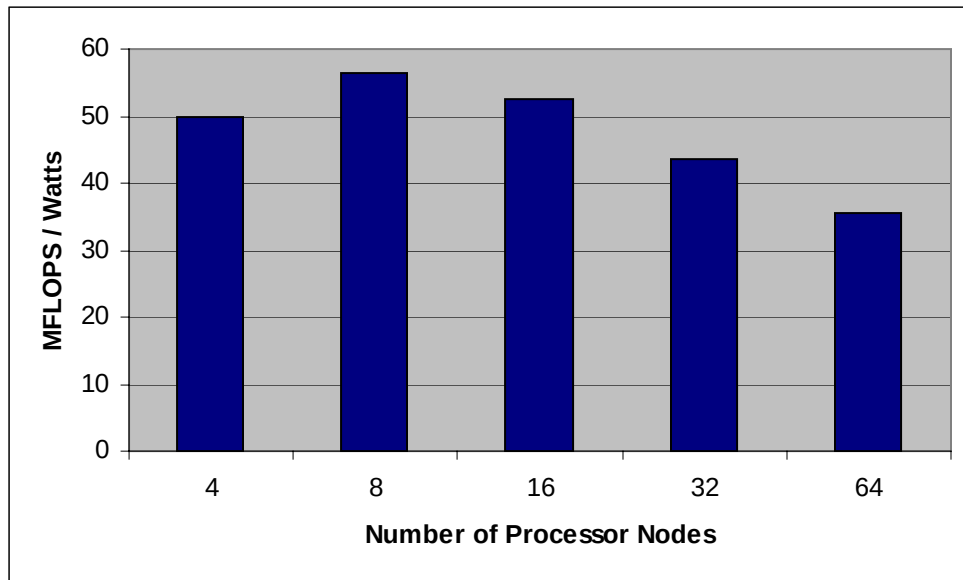


Figure 5.5: Energy Efficiency for Neighborhood

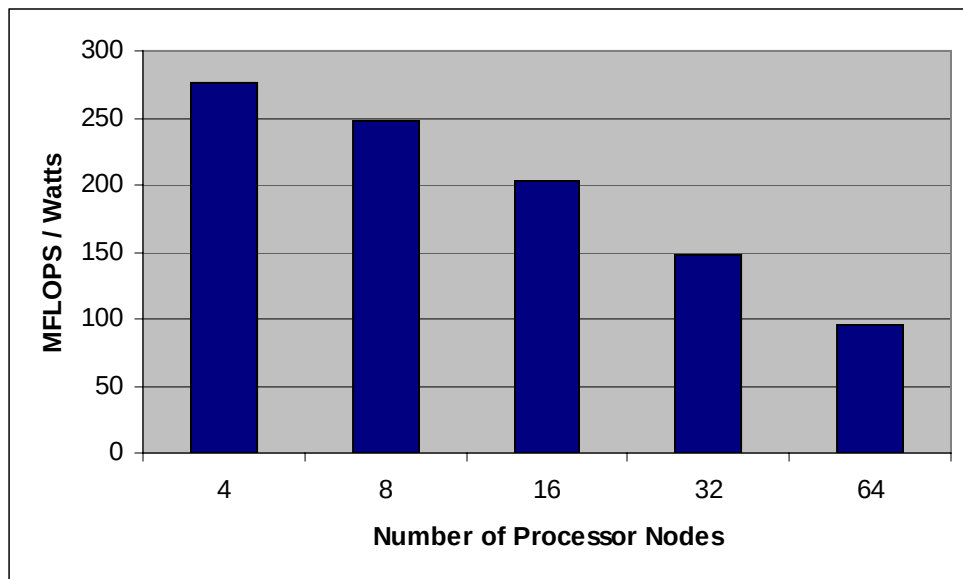


Figure 5.6: Energy Efficiency for Conjugate Gradient

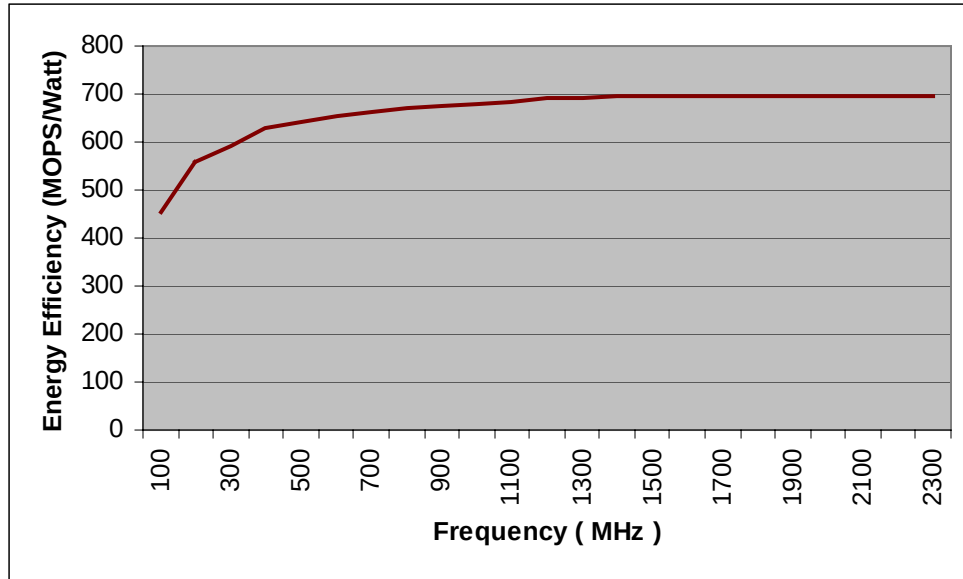


Figure 5.7: Energy Efficiency Vs Frequency (Transitive Closure)

5.4. Design Trade-offs

Simulation results show that power dissipation mainly depends on the frequency and the number of nodes. The distribution of power dissipation among the various hardware components shows that memory is the major contributor of power dissipation in the SCMP system. Figure 5.8 illustrates a typical distribution of power dissipation among the various subsystems in the SCMP system. Hence we can infer that power dissipation in SCMP mainly depends on three factors: the number of processor nodes, the clock frequency and the size of embedded DRAM. Each processor can have 1 to 8 MB of local memory. While some applications can efficiently run with only 1 MB of local memory, there are others that require more than 1 MB of memory. In order to design the system for an application, local memory of required size must be provided. In doing so, if the constraints in power dissipation are not met, then other factors that affect power dissipation have to be scaled. The choice and magnitude of this scaling must be arrived at after extensive analysis. For instance, if there is no significant performance degradation

by decreasing the number of processor nodes, a system consisting of lesser number of nodes, each with the required amount of DRAM can be implemented.

In Section 5.3, simulation results show an increase in power dissipation and performance with increasing number of nodes at a constant frequency of operation. Sometimes an increase in the number of nodes does not give a very good improvement in performance. This happens due to communication overheads. With increase in number of processor nodes computational work per processor decreases while the inter-processor communication increases. Significant increases in communication latency and the communication to computation ratio may counterbalance or outbid the performance gains obtained by the increase in TLP. As a result the expected improvement in performance may not be obtainable and in extreme cases performance may degrade. If performance improves with number of processor nodes but improves at a rate slower than power dissipation, it leads to degradation of energy efficiency. This is illustrated in simulation results in Section 5.3, where energy efficiency decreases in spite of the increase in performance for some benchmarks.

Power and performance are both directly proportional to frequency. The dependence of power dissipation and performance on frequency is well established. However the dependence of energy consumption and energy efficiency on frequency is revealed in the simulation results. The reason for this dependence was discussed in Section 5.3.2. At low frequencies of operation, static power dissipation is comparable to dynamic power dissipation and energy efficiency decreases with decrease in frequency. An immediate inference would be to increase the clock speed to the region where the energy efficiency curve flattens. However, it should be noted that it is not possible to arbitrarily increase clock speed beyond a certain limit without having to increase the level of pipelining, which will add more inter-stage registers and control logic and in turn increase the power dissipation due to pipeline overheads.

The optimal values for the memory size, number of nodes and frequency are dependent on the application, the required performance, energy efficiency and the power constraints. The choice of frequency, number of nodes and memory size are also dependant on other factors such as technology node, maximum operating frequency of the design and the die size. The ITRS power budget projections for the 70 nm, 50 nm and

35 nm technology nodes are shown in Table 5.1 [13]. The SCMP architecture can be used for high performance, cost performance, or hand held applications by choosing the memory size, clock frequency and number of nodes that give the required performance while meeting the power and energy constraints.

<i>Technology Node</i>	<i>70 nm</i>	<i>50 nm</i>	<i>35 nm</i>
<i>High-performance with heatsink (W)</i>	180	210	240
<i>Cost-performance (W)</i>	98	114	131
<i>Battery (W)—(low-cost/hand-held)</i>	2.4	2.7	3.0

Table 5.1: Maximum Allowable Power, ITRS Roadmap [13]

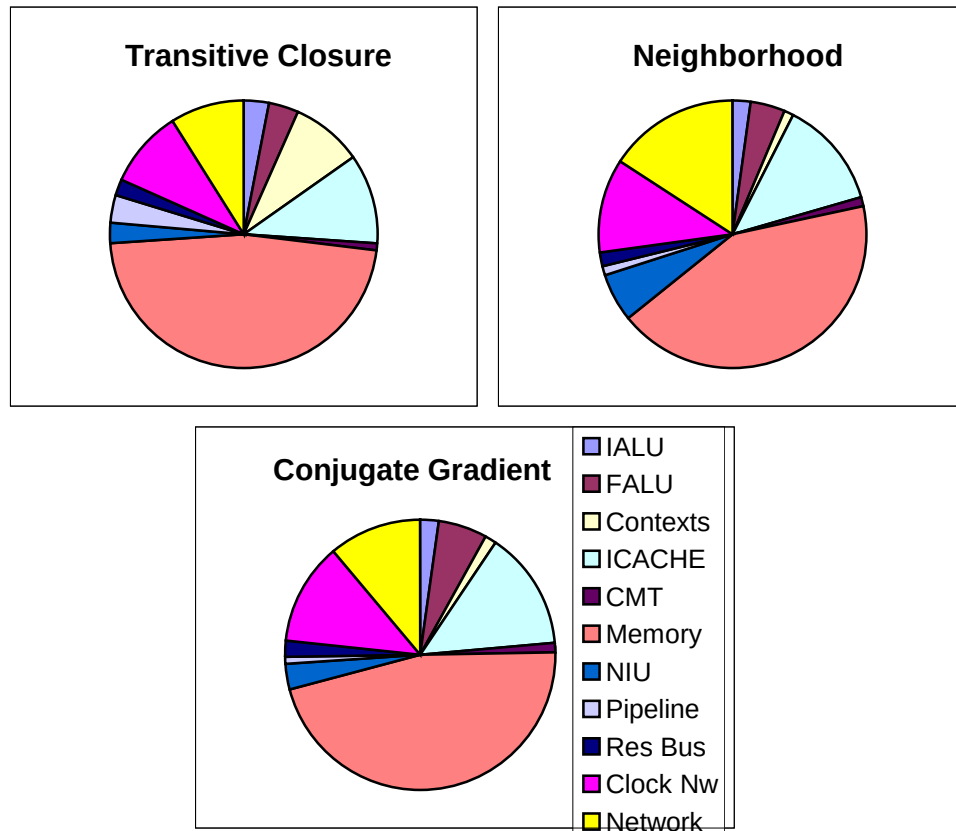


Figure 5.8: Distribution of Power Among Subsystems in SCMP

Chapter 6

6. Conclusions

6.1. Thesis Summary

In this thesis an infrastructure for microarchitectural level power analysis and optimization was proposed for Single Chip Parallel Computers. The power and performance simulator was developed and power optimizations were proposed based on the Single Chip Message-Passing Parallel Computer (SCMP). The power models for dynamic power dissipation and static power dissipation were developed for the SCMP system. Power dissipation in both the processor nodes and the on-chip network were modeled. Microarchitectural level details were included in a performance simulator in-order to include power estimation. Power models for the hardware structure were included in the microarchitectural level simulator by utilizing its modular structure. Several power analysis options were included in the simulator.

The power advantages inherent in the SCMP architecture targeted to DSM technologies were identified. Some power optimizations that reduce power dissipation without significant performance ramifications were proposed for the SCMP architecture. The power and performance analysis tool was used to analyze the power, performance and energy efficiency of the SCMP system across benchmarks from the Data Intensive Systems (DIS) stressmark suite. The design factors significantly influencing power dissipation were identified. The architectural level and system level design trade-offs involved in the design of the SCMP system were explored. The analysis and optimization presented in this thesis for the SCMP architecture can be extended to other Single Chip Parallel Computer architectures.

6.2. Future Work

Microarchitectural level power analysis is becoming important in exploring next generation computer architectures. This thesis aimed at developing a microarchitectural level power analysis tool for single chip parallel computers. Though significant work was completed in this thesis, there is room for future investigations.

In this work, the leakage power dissipation was estimated only for major components like array structures. Since leakage power is crucial for DSM designs, leakage power models for other hardware components should be developed. The subthreshold leakage current was modeled based on the BSIM3 transistor leakage equations. The BSIM4 equations are more accurate and up-to-date. Hence in future research leakage models should be developed based on the BSIM4 equations. The dynamic power models were developed for submicron technology nodes and were scaled to future DSM technology nodes based on projections from relevant research work. These dynamic power models should be validated by running some circuit level simulations.

The tool presented in this work is based on the SCMP system. Future work should concentrate on generalizing it for other single chip parallel computers. An important component required in the analysis of a new architecture is the delay analysis and the maximum clock speed achievable from a design. Since the power analysis tool was developed in close interaction with the SCMP hardware design team, an estimate of the achievable clock speed was available. In order to use the tool to analyze a completely new architecture or analyze the ramifications of significant architectural and system level changes inclusion of delay models will be helpful. The capacitance models developed for power analysis can be unutilized for delay analysis.

Bibliography

- [1] C. Belady, "Cooling and power considerations for semiconductors into the next century", *International Symposium on Low Power Electronics and Design*, pp. 100 – 105, 6-7 Aug. 2001.
- [2] C. Small, "Shrinking devices put the squeeze on system packaging", *Electronic Design News*, vol. 39, pp. 41 – 46, Feb 1994.
- [3] P. Yang and J. H. Chern, "Design for reliability: The major challenge for VLSI", *Proc. IEEE*, vol. 81, pp. 730 – 744, May 1993.
- [4] Anand Raghunathan, Niraj K. Jha, Sujit Dey, *High-Level Power Analysis And Optimization*, Kluwer Academic Publishers, Norwell, MA, 1998.
- [5] J. Rabaey and M. Pedram (Editors), *Low Power Design Methodologies*, Kluwer Academic Publishers, Norwell, MA, 1996.
- [6] Anantha P. Chandrakasan and Robert W. Brodersen, *Low power digital CMOS design*, Kluwer Academic Publishers, Norwell, MA, 1995.
- [7] Jerry Frenkil, "Tools and methodologies for low power design", *Proceedings of the 34th annual conference on Design Automation*, vol. 00, pp. 76 – 81, June 1997.
- [8] W. Nebel, J. Sproch, and S. Malik, "Power analysis and optimization: spanning the levels of abstraction", in *Tutorial Notes, Int. Symp. Low-Power Electronics and Design*, August 1997.
- [9] D. Sylvester and K. Keutzer, "Rethinking deep-submicron circuit design", *Computer*, vol. 32, Issue: 11, pp. 25 – 33, November 1999.
- [10] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron," *Proc. of ICCAD*, pp. 203-211, 1998.
- [11] D. Sylvester and K. Keutzer, "Getting to the bottom of deep submicron II: the global wiring paradigm," *Proceedings of International Symposium on Physical Design*, pp. 193-200, 1999.

- [12] R.Ho, K.W.Mai and M.A.Horowitz, "The future of wires", *Proc. of the IEEE*, Vol. 89, Issue. 4, pp. 490 – 504, April 2001.
- [13] "The International Technology Roadmap for Semiconductors", 2003 Edition, <http://public.itrs.net/Files/2003ITRS/Home2003.htm>
- [14] S. Manne, A. Klauser and D. Grunwald, "Pipeline Gating: Speculation Control For Energy Reduction", *Proc. Intl. Symp. Computer Architecture*, pp.132–141, June/July 1998.
- [15] K. Diefendorff and P. Dubey, "How Multimedia Workloads Will Change Processor Design", *Computer*, vol.30, no.9, pp.43-45, September 1997.
- [16] W.J. Dally and S. Lacy, "VLSI Architecture: Past, Present, and Future", *20th Conf. Advanced Research in VLSI*, pp.232-241, , March 1999.
- [17] K. Olukomn et al, "The case for a single chip multiprocessor", *ASPLOS-VII*, October 1996.
- [18] Karthikeyan Sankaralingam et al, "Exploiting ilp, tlp, and dlp with the polymorphous trips architecture", *International Symposium on Computer Architecture*, pp. 422–433, June 2003.
- [19] IBM Blue Gene Team, "Blue gene: A vision for protein science using a petaflop supercomputer", *IBM Systems Journal*, vol. 40, no. 2, pp. 310–327, 2001.
- [20] Elliot Waingold et al, "Baring it all to software: Raw machines", *Computer*, vol. 30, no. 9, pp. 86–93, 1997.
- [21] James M. Baker Jr. et al, "Scmp: A single-chip message-passing parallel computer", in *Parallel and Distributed Processing Techniques and Applications*, pp. 1485–1491, 2002.
- [22] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", *Proc. Intl. Symp. Computer Architecture*, pp.83-94, June 2000.
- [23] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling", *IEEE Computer Mag.*, vol. 35, pp. 59–67, Feb. 2002.

- [24] W. Ye et al, "The design and use of simplepower: a cycleaccurate energy estimation tool", in *Design Automation Conference*, pp. 340–345, 2000.
- [25] Y. Zhang, et al., "HotLeakage: An Architectural, Temperature-Aware Model of Subthreshold and Gate Leakage", *University of Virginia, Dept. of Computer Science Tech. Report CS-2003-05*, Mar 2003.
- [26] Hang-Sheng Wang, et al., "Orion: A Power-Performance Simulator for Interconnection Networks", *Proc. 35th Intl. Symp. Microarchitecture (MICRO-35)*, pp.294-305, November 2002.
- [27] Gary K. Yeap, *Practical Low Power Digital VLSI Design*, Kluwer Academic Publishers, Boston, MA, August 1997.
- [28] Abdellatif Bellaouar and Mohamed I. Elmasry, *Low-Power Digital VLSI Design - Circuits and Systems*, Kluwer Academic Publishers, Norwell, MA, June 1995
- [29] A. Agarwal, C. H. Kim, S. Mukhopadhyay, and K. Roy, " Leakage in Nanoscale Technologies: Mechanisms, Impact and Design Considerations", *Proceedings of the 41st annual conference on Design automation*, San Diego, CA, pp. 6 – 11, 2004.
- [30] Lei He, Weiping Liao, Mircea R. Stan, "System level leakage reduction considering the interdependence of temperature and leakage", *Proceedings of the 41st annual conference on Design automation*, San Diego, CA, Session: Hot leakage, pp.12-17, 2004.
- [31] J. A. Butts and G. S. Sohi, "A static power model for architects", *Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 191–201, December 2000.
- [32] U. C. Berkeley. BSIM3v3.2 SPICE MOS device models, June 16, 1998.
<http://www-device.eecs.berkeley.edu/~bsim/get.html>
- [33] U. C. Berkeley. BSIM4v2.1 SPICE MOS device models, October 5, 2001.
<http://www-device.eecs.berkeley.edu/~bsim3/bsim4.html>
- [34] W. Dally, J. Fiske, J. Keen, R. Lethin, M. Noakes, P. Nuth, R. Davison, and G. Fyler, "The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms", *IEEE Micro*, vol. 12, no. 2, pp. 23-39, April 1992.

- [35] D.S. Wills, H.H. Cat, J. Cruz-Rivera, W.S. Lacy, J.M. Baker, Jr., J.C. Eble, A. Lopez-Lagunas, and M. Hopper, "High-Throughput, Low-Memory Applications on the Pica Architecture", *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 10, pp. 1055-1067, October 1997.
- [36] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, 2nd ed, New York: Addison Wesley, 1994.
- [37] M. Borah, R.M. Owens, M.J Irwin, "Transistor sizing for low power CMOS circuits", , *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15 , no. 6 , pp. 665 – 671, June 1996
- [38] R. Zimmermann, W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic", *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, pp. 1079 – 1090, July 1997.
- [39] Hang-Sheng Wang, Li-Shiuan Peh, S. Malik, "A power model for routers: modeling Alpha 21364 and InfiniBand routers", *IEEE Micro*, vol. 23, no. 1, pp. 26 – 35, Jan.-Feb.2003
- [40] Brian Gold, *Balancing Performance, Area, And Power In An On-Chip Network*, Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, July 2003
- [41] "National Technology Roadmap for Semiconductors", Semiconductor Industry Association, 1997.
- [42] Ron Ho, Ken Mai, M. Horowitz, "Managing wire scaling: a circuit perspective", *Proceedings of the IEEE 2003 International Interconnect Technology Conference*, pp. 177 – 179, 2-4 June 2003.
- [43] V. Tiwari, et al., "Reducing Power in High-performance Microprocessors", *Proc. Design Automation Conference*, pp.732-737, June 1998.
- [44] A. Hemani, et al., "Lowering power consumption in clock by using globally asynchronous locally synchronous design style", *Proc. Design Automation Conference*, pp.873-878, June 1999.
- [45] R. Viswanath, V. Wakharkar, A. Watwe and V. Lebonheur, "Thermal Performance Challenges from Silicon to Systems", *Intel Technology Journal Q3*, 2000.

- [46] R. Mahajan, K. Brown, V. Atluri, “The Evolution of Microprocessor Packaging”, *Intel Technology Journal Q3*, 2000.
- [47] V. Zyuban and P. Kogge. “Split register file architectures for inherently lower power microprocessors”, *Proc. Power-Driven Microarchitecture Workshop, in conjunction with ISCA'98*, pp.32-37, 1998.
- [48] J.-L.Cruz, et al., “Multiple-banked register file architectures”, *Proc. 27th Intl. Symp. Computer Architecture*, pp.316–325, June 2000.
- [49] M. Powell, et al., “Gated-Vdd: A circuit technique to reduce leakage in cache memories”, *Proc. Intl. Symposium on Low Power Electronics and Design*, pp.90-95, July 2000.
- [50] DIS Stressmark Suite, v1.0. Titan Systems Corp., 2000.
<http://www.aaec.com/projectweb/dis/>
- [51] R. Gonzalez and M. Horowitz, “Energy Dissipation in General Purpose Microprocessors”, *IEEE J. Solid-State Circuits*, vol.31, no.9, pp.1277–1284, September 1996.
- [52] B.R. Gaeke, et al., "Memory-Intensive Benchmarks: IRAM vs. Cache-Based Machines," *Proc. Intl. Parallel and Distributed Processing Symp. (IPDPS)*, pp.30-36, April 2002.
- [53] B. Nadel, “The Green Machine”, *PC Magazine*, vol. 12, May 1993.

Vita

Priyadarshini Ramachandran received her Bachelor's degree in Electronics and Communication Engineering from the Bharathiar University in Coimbatore, India, in May, 2002. In August, 2002 she joined Virginia Tech to pursue her Master of Science in Electrical Engineering. During the course of her graduate studies she was a Research Assistant under Dr. James M. Baker. She received her M.S.E.E. in July, 2004. She joined National Instruments in August 2004 and now serves as a Digital Design Engineer. Her research interests include low power VLSI design, computer architecture and CMOS digital circuit design.