Methodologies and Simulation
Framework for Architectural Analysis
and Power Management of FinFET Chip
Multiprocessors

Chun-Yi Lee

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Electrical Engineering
Adviser: Professor Niraj K. Jha

April 2013
© Copyright by Chun-Yi Lee, 2013.

All rights reserved.
Abstract

This dissertation presents a simulation framework, called FinCANON, for modeling power and timing to support a comprehensive design space exploration of caches and networks-on-chip (NoCs) in a chip multiprocessor system. It also describes methodologies for analyzing the impact of process, voltage, and temperature (PVT) variations on power consumption and delay. At the architecture level, it introduces a flow control mechanism to manage both the throughput and power consumption of NoCs.

FinFETs have emerged as promising substitutes for bulk CMOS at the 22nm technology node and beyond. Nevertheless, PVT variations in FinFETs lead to large spreads in delay and leakage. We have developed a FinFET design library to model the circuit-level characteristics as well as their variation trends with respect to various PVT parameters for FinFET logic gates and memory cells. Based on a statistical static timing analysis technique and macromodel based methodology, we have derived the PVT variation models for delay and leakage, taking into account spatial correlations, to characterize the impact of PVT variations on FinFET-based caches and NoCs.

Based on the FinFET design library, we next present FinCANON, an integrated framework for the simulation of power, delay, as well as PVT variations of FinFET-based caches and NoCs. FinCANON is built atop CACTI-PVT and ORION-PVT that model caches and NoCs, respectively. FinCANON enables architects to evaluate the impact of PVT variations on caches and NoCs at an early design stage. We present results for various FinFET design styles and show that mixing different styles may be a promising strategy for optimizing delay and leakage of caches and NoCs.

We next discuss the microarchitecture and flow control mechanism of a variable-pipeline-stage router (VPSR). VPSR adjusts the number of pipeline stages based on incoming traffic to a router port, leading to significant savings in leakage power while
maintaining router throughput. We also propose enhanced token flow control, a flow control mechanism that improves upon the energy-delay-throughput of the previous state-of-the-art token flow control mechanism. We propose a new concept of using guaranteed tokens to establish temporary express virtual channels to quickly bypass packets from congested regions.
Acknowledgments

First and foremost, I would like to express my heartfelt gratitude to Professor Niraj K. Jha for all his help, advice, and support over the past five years. I thank Professor Jha for leading me into the brave new FinFET chip multiprocessor world. He has always been patient with me and given me the freedom to explore new ideas. In addition to supervising my Ph.D. research, I would like to thank Professor Jha for teaching me invaluable lessons on tackling problems, presentation skills, meditation, article writing, and advice for life and career. I still have a lot to learn from his wisdom, discipline, foresight, and patience. I wish that I can be the kind of great advisor like him, and be able to guide young generations in the future.

I would like to thank Professor Naveen Verma and Professor David Wenzlaff for being my thesis readers, and Professor Ruby Lee and Professor Sun-Yuan Kung for agreeing to be on my Ph.D. defense committee.

I am deeply grateful to Professor Sun-Yuan Kung, who was my academic advisor during my first semester and a committee member in my general exam, for his advice and guidance. In addition, I am also grateful for the valuable discussions with Professor Wei Zhang. I would like to thank both of them for their insightful comments and constructive suggestions at different stages of my Ph.D. study.

I would also like to show my gratitude to other members of my research group: Yang Yang for his help with generating the FinFET logic library and developing the PVT variation models, Sourindra Chaudhuri for providing his 2D TCAD FinFET models, Xianmin Chen for establishing the GEM5-FinFET infrastructure, Aoxiang Tang and Ajay Bhoj for help with FinFET caches and SRAMs, Shoaib Mohammed and Amlan Chakrabarti for enduring me during our countless conversations, and Ting-Jung Lin, Chia-Chun Lin, Chunxiao Li, Jun Wei Chuah, Meng Zhang, and Debajit Bhattacharya for their support, great friendship, and valuable feedback on my work.
I would like to thank my roommates Hau-Tieng Wu, Yu-Chao Tu, Michael Wang, Lawrence Wang, and my friends Jakub Szefer, Yu-Yuan Chen, Carole-Jean Wu, An Dai, Arnab Sinha, Niket Agarwal, Yu-Cheng Tsai, Amy Wu, Cheng-Hsien Li, Liang-Yen Hsu, Yae-Lin Sheu, Yen-Wei Chang, Kung-Ching Liao, Ya-Chin Chiou, Debbie Faigen, Mickey Faigen, Charlie Shucheng Zhu, Chia-Hsin Owen Chen, and Pei-Yuan Wu, for their physical and emotional support, and enlightening me with interesting research ideas. I would also like to thank my manager Shriram Gundala at Oracle for his support and help.

Finally, my deepest thanks to my parents Tsung-Ming Lee and Yueh-Fen Yang, and my sister Yu-Chun Lee, who gave me so much support and understanding throughout my whole life. None of this work would have been possible without the love and patience of my family.
To my parents and my sister, who have supported me throughout my education.
Contents

Abstract ........................................................................................................ iii
Acknowledgments ......................................................................................... v
List of Tables ................................................................................................. xii
List of Figures ............................................................................................... xiii

1 Introduction ............................................................................................... 1
  1.1 CMP Architectures ............................................................................... 3
    1.1.1 On-chip networks ......................................................................... 4
    1.1.2 Caches in CMPs ........................................................................... 8
  1.2 CMP Power Consumption ...................................................................... 9
  1.3 CMOS Scaling Challenges .................................................................... 10
  1.4 FinFETs ............................................................................................... 11
  1.5 PVT Variations .................................................................................... 15
  1.6 Dissertation Contributions ................................................................... 17
    1.6.1 FinFET design library ................................................................. 18
    1.6.2 FinCANON ................................................................................ 19
    1.6.3 Variable-pipeline-stage router ................................................... 20
    1.6.4 Enhanced token flow control ....................................................... 21
  1.7 Dissertation Structure .......................................................................... 22

2 Related Work ............................................................................................. 23
# 2. Cache Modeling

- 2.1 Cache Modeling ................................................. 23
- 2.2 On-chip Network Modeling ....................................... 25
- 2.3 Processor Modeling .............................................. 27
- 2.4 CMP Performance Simulation .................................... 30
- 2.5 PVT Variation Modeling ......................................... 31
- 2.6 NoC Flow Control .............................................. 34
  - 2.6.1 Credit-based flow control mechanism ...................... 34
  - 2.6.2 Express virtual channel .................................. 34
  - 2.6.3 Token flow control ...................................... 35

# 3. FinFET Design Library ............................................. 37

- 3.1 Introduction ................................................... 37
  - 3.1.1 FinFET design library construction ................. 38
- 3.2 PVT Variation Models .......................................... 40
  - 3.2.1 Delay variation model .................................. 42
  - 3.2.2 Leakage variation model ................................ 46
  - 3.2.3 Temperature macromodel ............................... 48
- 3.3 Voltage Generators ............................................ 49
- 3.4 Experimental Results .......................................... 51
  - 3.4.1 Simulation setup ........................................ 51
  - 3.4.2 Impact of PVT variations on FinFET logic gates ........ 52
  - 3.4.3 Validation of delay and leakage models .............. 58
  - 3.4.4 Comparison of FinFET logic gate styles .............. 59
  - 3.4.5 Leakage power of FinFET memory cells ............... 61
  - 3.4.6 Impact of PVT variations on FinFET circuits ........ 61
- 3.5 Chapter Summary .............................................. 65
4 FinCANON Simulation Framework

4.1 Introduction .................................................. 69
  4.1.1 Architecture-level macromodels ......................... 71
4.2 CACTI-PVT ..................................................... 72
  4.2.1 Mat organization ........................................ 75
4.3 ORION-PVT ..................................................... 75
  4.3.1 Delay and power models of NoC components .............. 78
4.4 GARNET-FinFET ............................................... 81
4.5 Experimental Results ......................................... 82
  4.5.1 Cache delay and leakage characterization .............. 84
  4.5.2 Cache delay and leakage vs. $3\sigma/\mu$ variations ... 86
  4.5.3 Delay and leakage vs. temperature .................... 87
  4.5.4 Comparison of FinFET operation modes ................. 87
  4.5.5 Delay and leakage vs. cache capacity .................. 91
  4.5.6 Delay and leakage vs. memory styles ................... 91
  4.5.7 Delay and power of router components .................. 94
  4.5.8 Network power breakdown ................................ 95
  4.5.9 Power comparison for various types of traffic patterns .... 96
  4.5.10 Network power vs. temperature ....................... 98
  4.5.11 PARSEC simulations ................................... 98
4.6 Chapter Summary ............................................. 99

5 Variable-pipeline-stage Router ................................ 102

5.1 Introduction .................................................. 102
5.2 Critical Path Analysis and Optimization .................... 103
5.3 Variable Pipeline Stages .................................... 104
  5.3.1 Flow control mechanism ................................. 106
5.4 VPSR Dynamic Power Management (VPSRDPM) ................ 106
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>22nm FinFET design parameters</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Variable PVT parameters</td>
<td>51</td>
</tr>
<tr>
<td>3.2</td>
<td>Impact of PV variations on $C_{in}$</td>
<td>56</td>
</tr>
<tr>
<td>3.3</td>
<td>Impact of PV variations on $C_{out}$</td>
<td>56</td>
</tr>
<tr>
<td>3.4</td>
<td>Impact of PV variations on $R_{out}$</td>
<td>57</td>
</tr>
<tr>
<td>3.5</td>
<td>Impact of PV variations on $I_{Leak}$</td>
<td>57</td>
</tr>
<tr>
<td>3.6</td>
<td>Residual and statistical errors of the delay model</td>
<td>59</td>
</tr>
<tr>
<td>3.7</td>
<td>Residual and statistical errors of the leakage model</td>
<td>60</td>
</tr>
<tr>
<td>3.8</td>
<td>Comparison of FinFET logic gate styles</td>
<td>60</td>
</tr>
<tr>
<td>3.9</td>
<td>Impact of temperature variation on $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ for s38584</td>
<td>63</td>
</tr>
<tr>
<td>3.10</td>
<td>Validation of the PVT variation models</td>
<td>66</td>
</tr>
<tr>
<td>4.1</td>
<td>Cache configuration parameters</td>
<td>83</td>
</tr>
<tr>
<td>4.2</td>
<td>Network parameters</td>
<td>83</td>
</tr>
<tr>
<td>4.3</td>
<td>Variable PVT parameters</td>
<td>84</td>
</tr>
<tr>
<td>4.4</td>
<td>Delay of router components</td>
<td>95</td>
</tr>
<tr>
<td>4.5</td>
<td>Leakage power of router components</td>
<td>95</td>
</tr>
<tr>
<td>5.1</td>
<td>Network parameters</td>
<td>124</td>
</tr>
<tr>
<td>5.2</td>
<td>Network latency (cycles) under two routing schemes</td>
<td>128</td>
</tr>
<tr>
<td>5.3</td>
<td>Network latency (cycles) under different input buffer sizes</td>
<td>129</td>
</tr>
</tbody>
</table>
5.4 Network latency (cycles) under different numbers of VCs
List of Figures

1.1 CMP examples .............................................. 4
1.2 CMP structure ............................................. 5
1.3 Baseline router microarchitecture [23, 24] ..................... 6
1.4 Baseline router pipeline stages [12] .......................... 6
1.5 FinFET structure and modes .................................. 13
2.1 Cache structure in CACTI [100] ................................. 24
2.2 Router power breakdown for bulk CMOS ........................ 27
2.3 Modeling hierarchy of McPAT [33] ............................... 30
3.1 FinFET design library construction flow ......................... 38
3.2 Parameter extractor block diagram ............................. 39
3.3 Multi-level grid-based method .................................. 41
3.4 Voltage generator design ..................................... 50
3.5 FinFET logic gate layouts [88] .................................. 52
3.6 FinFET SRAM cells .......................................... 53
3.7 QMC parameter characterization of s27 ......................... 54
3.8 Impact of temperature variation on $C_{in}$, $C_{out}$, $R_{out}$, and $I_{Leak}$ ....... 58
3.9 Memory cell leakage vs. temperature ........................... 62
3.10 Impact of PV variations on $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ for s38584 ($T_{op} = 298K$) 64
3.11 STD errors of $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ for s38584 ($T_{op} = 298K$) ....... 67
5.11 VPSR microarchitecture .................................................. 121
5.12 Average network latency vs. injection rate ............................ 125
5.13 Power consumption for various types of network traffic .......... 127
5.14 Power breakdown for the tornado traffic ............................. 130
5.15 Network power for the PARSEC traffic .............................. 131

6.1 McPAT-PVT macromodel generation flow .............................. 139
Chapter 1

Introduction

Chip multiprocessors (CMPs) have become the mainstream high-performance processor architecture. In modern CMP designs, caches and networks-on-chip (NoCs) have become major components [1 2 3 4 5 6 7 8]. With increasing demand for cache capacity and density, state-of-the-art CMP designs dedicate a large fraction (40%~60%) of die area to caches [9 10]. Another trend in modern CMP designs is the use of NoCs. To meet the increasing performance requirements, NoCs have emerged as a popular design paradigm for CMPs and non-uniform cache architectures (NUCAs). NoCs connect processor cores, caches, as well as memory controllers in CMPs, and regulate packet communications in its network [11 12]. Caches and NoCs also work synergistically to provide cache coherence in modern CMP designs [4 5 6 7 8]. However, as technology moves into the deep-submicron regime, shrinking feature size has placed considerable stress on CMOS fabrication due to short-channel effects (SCEs), quantum-mechanical tunneling of carriers through thin gate oxide, decreasing supply and threshold voltages, increasing channel doping, lithography and mask constraints, and excessive leakage [13 14 15]. Although many research efforts have been devoted to seeking system-level solutions [16 17], underlying transistor-level solutions are still urgently required to overcome these obstacles. Additionally, because of pack-
aging and heat dissipation constraints, as well as ever-increasing power densities and cooling costs, it has become important to estimate the power consumption of each component both at an early design stage as well as during the execution of an application on a CMP. Moreover, circuit performance at and beyond the 22nm technology node is extremely sensitive to variations in manufacturing processes, supply voltage, and temperature (PVT). Process variations result in a non-uniformity of transistor delay and leakage power across a single die, and can cause a difference in chip frequency and leakage from die to die by as much as 30% [18] and $20\times$ [19], respectively. Supply voltage fluctuations lead to delay and power uncertainty [20]. The delay and leakage of logic gates and memory cells are further affected by temperature variations both in time and space. If not properly addressed, PVT variations in CMPs produce collision of packets, packet losses, increased access latency, and hot spots. Such variations affect CMP design metrics as well, including performance, power consumption, reliability, and cost. As a result, considering PVT variation issues at an early design stage of a high-performance CMP has now become necessary to ensure a reasonable yield in fabrication. An important step in addressing this issue is understanding how PVT variations affect delay and power in CMPs. This requires fast and accurate PVT variation analysis tools. Finally, since high-end applications running on CMPs require delivery of maximum performance under a given stringent power budget, it is important to estimate and optimize the power consumption of nodes in CMPs. Due to the high cost of worst-case design (to tackle worst-case power and temperature), it is also important to apply dynamic power management (DPM) techniques to CMPs to alleviate power emergencies at runtime.
1.1 CMP Architectures

In the area of processor architectures, the trend in the past few years has moved from uniprocessor designs to CMPs. In the era of uniprocessors, applications with different resource requirements, e.g., computationally intensive applications, memory/cache access-intensive applications, and applications that require parallel processing of instructions and data, were handled in a single processor. To achieve high instruction throughput, techniques such as pipelining, dynamic scheduling, out-of-order processing, and multithreading have been extensively used. These techniques usually equip microprocessors with large caches and multiple arithmetic-logic/floating-point units (ALUs/FPUs). However, as the number of resource-intensive applications running on a uniprocessor increases, uniprocessor architectures face hurdles in improving instruction throughput. Additionally, CMOS technology is approaching its limit of operation frequencies. Uniprocessors can rarely reach frequencies beyond 4GHz. Moreover, complex processor designs lead to higher power consumption. CMPs have emerged as an effective means for utilizing MOSFETs to continue the growth of instruction throughput of microprocessors. A state-of-the-art CMP is typically composed of multiple processing elements (PEs) interconnected with a network fabric and a large amount of on-chip caches. CMP architectures have three advantages: throughput improvement, dynamic power/temperature management, and heterogeneous architectures. First, different applications can be assigned to different groups of PEs, leading to higher instruction/data parallelism and thus throughput. Second, different PEs can run at different frequencies and supply voltages, allowing power consumption and temperature to be dynamically managed. Third, the PEs connected by the network fabric can be either homogeneous or heterogeneous. This enables a number of different types of PEs, e.g., graphics processing units (GPUs), field-programmable gate array (FPGAs), and application-specific integrated circuits (ASICs), to be incorporated into a CMP system to speed up critical sections of the application and improve
Figure 1.1: CMP examples

power efficiency [21]. It is expected that the number of PEs per compute node will keep growing towards hundreds and possibly a thousand. A number of CMP architectures have been proposed by both academia [1, 2, 3] and industry [4, 5, 6, 7, 8]. Fig. 1.1 shows some of the state-of-the-art CMP designs. The network fabric, which connects the PEs, can be a dedicated bus, point-to-point network, or an NoC. Due to the availability of sufficient on-chip wires, inter-PE links are not constrained by chip pin counts. Hence, high bandwidth as well as performance can be achieved. Bus-based interconnection has limitation in its scalability, because only one PE at a time can utilize the bus. On the other hand, point-to-point communications require a massive number of interconnects as the number of PEs grows. As a result, efficient NoC design has been attracting increasing attention in recent years.

1.1.1 On-chip networks

With increasing wiring delay and demand for high bandwidth [22], packet-switched on-chip networks have been recognized as a scalable and flexible solution for on-chip
communications in CMPs [11]. The concept has evolved from modern computer networks. An on-chip network connects PEs and caches through routers and links. New PEs or cache banks can be easily added to a CMP design by connecting them to the on-chip network, which greatly enhances scalability. Messages are exchanged between PEs through packets. Packets are further divided into fixed-length flow control units (flits) to efficiently use on-chip router resources and enable a number of routing algorithms. A structured on-chip network also enables the use of shorter wires, which reduces packet traversal latency.

Fig. 1.2 shows a CMP structure, which consists of PEs, cache banks, and a 4×4 network. The PEs and cache banks are connected to the network by on-chip routers, which transfer packets between the processors and caches. The components connected to the network vary for different designs and applications. If most of them are PEs, it forms a high-performance multi-processor system. If most of them are cache banks, it forms a non-uniform cache architecture (NUCA) system, which is described in the next subsection.
The function of a router is to route flits from the input ports to their requested output ports. A packet typically consists of three types of flits: the header flit, body flit, and tail flit. The header flit contains the address of the destination PE and sometimes the routing information. The body flit consists of the contents of the packet. The tail flit is the end of the packet, which is usually used to release the reserved router resources. Flits injected into a network must compete for router pipeline resources on a hop-by-hop basis along the path from their source nodes to destination nodes. Therefore, the number of router pipeline stages dominates both flit latency and router power consumption.

Figs. 1.3 and 1.4 show the microarchitecture [23, 24] and pipeline stages [12] of the baseline router. As a flit arrives at the input port of the router, it is first written
into the input buffer in the buffer write (BW) stage. In parallel, a route computation (RC) unit determines the output port where the flit is to be forwarded. If the flit is the header of a packet, it goes through the virtual channel allocation (VA) stage and tries to acquire a free virtual channel (VC). Next, in the switch allocation (SA) stage, the switch allocator determines which flits have the right to go through the crossbar switch. The SA stage consists of a local switch arbiter (LSA) and a global switch arbiter (GSA). The LSA arbitrates among VCs and selects one VC for each input port. The GSA then determines which winning VC has the right to access the crossbar. The flit from the winning VC is then read out and traverses the crossbar in the switch traversal (ST) stage. Finally, the flit travels to the downstream router in the link traversal (LT) stage. In this dissertation, we assume the microarchitecture and pipeline stages illustrated in Figs. 1.3 and 1.4, respectively. However, note that some CMPs use only one or two pipeline stages in their routers [7, 8].

Pipeline bypassing is a commonly used technique to aggressively shorten the router pipeline. It allows a flit to traverse only the ST and LT stages, leading to a significantly shorter network latency [23]. A lookahead (LA) is sent to the next hop one cycle in advance to set up the crossbar switch for the desired output port. Flit bypassing does not require buffer read and write, thus consumes less energy in the router. Another benefit of flit bypassing is that more free buffers are available for other packets, thereby improving throughput at high traffic loads. Flit bypassing was previously exploited in [24, 25, 26] under very low traffic loads where flits do not need to compete for router resources with each other. In [24], the authors expand the concept of flit bypassing to an express virtual channel (EVC) technique, which allows packets to virtually bypass intermediate routers along pre-defined virtual express paths between pairs of nodes. In [23], the authors further generalize the concept of flit bypassing to a token flow control (TFC) mechanism. TFC regulates flit bypassing through the use
of tokens, and approaches the energy-delay-throughput of dedicated wires. EVC and TFC are described later.

1.1.2 Caches in CMPs

NUCAs have been widely adopted in CMPs and extensively studied [28, 29, 30, 31, 32]. As opposed to the uniform cache architecture (UCA), which is commonly used in uniprocessors, a NUCA allows large caches to be divided to smaller ones and distributed in an on-chip network. The distributed cache banks can be inside PEs, or can form cache tiles in a network. The access latency to each cache bank varies based on the distance from the requesting node, which is calculated as the sum of network delay and the cache access delay. A NUCA provides the benefits of both private and shared cache architectures. Each PE keeps a portion of its caches as private (L1/L2) and the rest as shared (L2/L3). Frequently used data are moved to the requesting node’s private portion or the shared portion of its neighborhood. As a result, shorter access latency can be achieved. Additionally, NUCAs provide higher access bandwidths than UCAs due to multiple distributed cache banks. NUCAs also allow migration of cache contents, which enables access efficiency and a number of dynamic thermal management (DTM) schemes. Few simulation tools are available for simulating NUCAs. The most popular one is CACTI 6.5 [33], which includes a detailed model for CMOS-based caches and simple models for on-chip routers and links. CACTI models CMOS-based caches using a CMOS technology library. The technology library contains device-level parameters, such as gate length, oxide thickness, threshold voltage, etc., for various technology nodes. Circuit-level parameters, such as capacitances and resistances of logic gates and memory cells, are derived from CACTI’s analytical models and device-level parameters in its technology library. The delay and power of CMOS-based caches are then derived from the circuit-level parameters of logic gates and memory cells.
1.2 CMP Power Consumption

A continuous increase in the number of processor tiles and demand for high-bandwidth data communication in CMPs have led to a significant increase in the power consumed by on-chip packet-switched interconnection networks. A number of CMP designs have been shown to dissipate significant power in their on-chip network fabric. For instance, in the Alpha 21364 processor, the integrated routers and off-chip links consume 20% of its total power [34]. The peak power from routers and links in Intel’s teraflop processor is about 28% of the tile power [3, 35]. High power consumption increases temperature, leading to hotspots and, hence, degrades performance and reliability. Moreover, it increases the cost of cooling IC packages. Hence, DPM is urgently required for interconnection networks to meet their constrained power budgets [36, 37].

In addition to power consumption, the state-of-the-art on-chip packet-switched network incurs significant delay and energy overhead due to the complex router pipeline stages. Packet latency and energy are mainly dominated by contention at the router pipeline due to resource competition. As a result, it is necessary to close the gap between packet-switched networks and the ideal (though impractical) interconnect fabric in which each pair of nodes is connected by a dedicated wire.

Another significant challenge we face is the increasing role played by leakage power in the overall CMP power consumption, not just in the idle mode, but also in the active mode. The active-mode leakage power in CMOS circuits was estimated to be as much as 40% of the total power consumption at the 70nm technology node [38]. Although leakage current can be controlled by engineering at the device level, it continues to increase as the feature size scales down [22]. As transistors scale down to deep sub-micron technology nodes, leakage is becoming the dominant part of power consumption [22]. Moreover, various scaling obstacles are faced by bulk CMOS, such as SCEs, process variations, dopant fluctuations, etc. Although circuit-level techniques like reverse body-biasing (RBB) [39] and adaptive body-biasing (ABB) [40]
have been proposed to reduce standby and active-mode leakage, respectively, it has been reported in [42] that the role of well/body bias in threshold voltage modulation is becoming less effective as CMOS scales down. Traditionally, cooling system design has been based on worst-case analysis. As heat density and system complexity scale further, worst-case design becomes increasingly difficult and infeasible. This has led to recent processor designs [44, 45] moving to average-case thermal design and employing DTM schemes upon occurrence of thermal emergencies.

The rapid increase in processor cache size has also led to a significant increase in the power consumed by on-chip caches [44, 45]. This power is mainly due to the dynamic power of cache accesses and leakage power of memory cells in standby mode. The dynamic power can be reduced by taking advantage of the fact that not all the banks are frequently accessed [45, 46, 47]. Thus, only a limited number of banks may be allowed to remain active while disabling the rest of the banks. However, low-power cache designs have started focusing on leakage power because of its increasing importance [48, 49, 50, 51, 52]. The concept of drowsy cache [48, 49, 53] allows cold cache lines to be placed into a state-preserving, low power drowsy mode. The cache decay concept [54] invalidates and turns off cache lines when their data content is not likely to be used in the near future. Power density minimization techniques mitigate thermal effects [45] by turning off alternate cache lines.

1.3 CMOS Scaling Challenges

Over the past two decades, CMOS scaling has consistently scaled well and driven the growth of high-density, high-speed, and low power VLSI systems. Continuous shrinking of transistor sizes with new generations of bulk CMOS technologies has yielded continual improvement in the performance of microprocessors [55]. The scaling of bulk CMOS, however, faces significant challenges due to fundamental material
and process technology limits. According to the International Technology Roadmap for Semiconductors (ITRS) [22], major obstacles to the scaling of bulk CMOS to sub-22nm gate lengths include quantum-mechanical tunneling of carriers through thin gate oxide, SCEs, control of the density and location of dopants, subthreshold leakage, finite subthreshold slope, and device-to-device variations. Even though high-k dielectrics have been used to control the gate leakage, issues such as thermal stability, inter-facial layer control, equivalent oxide thickness (EOT) control, reactions with polysilicon, and metal gate electrodes are still posing varying degrees of challenges [56]. Therefore, new underlying transistor-level solutions are needed to overcome the above problems.

### 1.4 FinFETs

FinFETs have emerged as promising substitutes to bulk CMOS at the 22nm node and beyond due to their better scalability, offering an opportunity to sustain Moore’s law [22]. FinFET is a type of non-planar double-gate device that provides tighter control of SCEs, reduced subthreshold leakage, and better scalability [57, 58, 59]. Researchers have demonstrated these benefits through FinFET 4T/6T/8T SRAM cells [42, 60, 61, 62] and embedded-DRAM cells [63], which have been shown to have significantly reduced leakage power as compared to CMOS SRAM cells. In [64], researchers have shown that FinFET 6T SRAM cells consume only 7.56% of the leakage power consumed in CMOS 6T SRAM cells. Additionally, independent control of the two transistor gates in FinFETs enables a number of creative circuit modules [65, 66, 67, 68, 69] and DPM/DTM schemes [70, 71]. FinFETs also enable higher transistor density, because fin height determines effective channel width. Intel has commercialized its first FinFET processor at the 22nm node in 2012.
Fig. 1.5(a) shows the geometry of a FinFET device, which employs a thin fin as the channel body. The gate length \((L_G)\) is equal to the fin length. The fin thickness \((T_{ST})\) is chosen to be small relative to \(L_G\), in order to ensure that the gate has effective control over the channel. FinFETs provide a number of advantages, such as shorter delay, higher on-state versus off-state current ratio \((I_{ON}/I_{OFF})\), and much less leakage than traditional bulk CMOS. The use of a lightly-doped channel makes FinFETs resistant to channel dopant fluctuation. Fig. 1.5(b) shows the cross-section of a FinFET. Here, \(L_{GF}, L_{GB}, T_{OXF}, T_{OXB}, H_{GF}, H_{GB}, L_{SPF}, L_{SPB},\) and \(L_{UN}\) denote physical front- and back-gate lengths, front- and back-gate effective oxide thicknesses, front- and back-gate thicknesses, front- and back-gate spacer thicknesses, and gate-drain/source underlap, respectively.

Shorted-gate (SG), independent-gate (IG) \([65]\), and asymmetric-workfunction shorted-gate (ASG) \([72]\) modes are three possible ways to implement FinFET logic gates. Fig. 1.5(c) illustrates SG-, IG-, and ASG-mode inverters, respectively. In the SG mode, the front and back gates of a FinFET are tied together. In the IG mode, the back gate of the pFinFET (nFinFET) is reverse-biased to a separate \(V_{HI} (V_{LOW})\). The threshold voltage \((V_{th})\) of the front gate varies linearly with the bias voltage. This allows the leakage current of an IG-mode gate to be reduced by one-to-two orders of magnitude. A higher reverse bias leads to much less leakage current, but increases the gate delay due to the higher \(V_{th}\). IG-mode FinFETs provide the advantage of controlling \(V_{th}\), however, occupy larger layout area due to the extra contacts required for the back-gates. For SG and IG modes, the front and back gates of a FinFET share the same workfunction \((\Phi_G)\). In the ASG mode, the front and back gates are tied together as in the SG mode, but have different workfunctions. ASG-mode FinFETs provide ultra-low off-currents, lower than even IG-mode FinFETs, and relatively high on-currents, though not as high as SG-mode FinFETs. ASG-mode FinFETs also occupy the same layout area as SG-mode
Figure 1.5: FinFET structure and modes
Table 1.1: 22nm FinFET design parameters

<table>
<thead>
<tr>
<th>FinFET parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fin height, $H_{FIN}$ (nm)</td>
<td>40</td>
</tr>
<tr>
<td>Gate length, $L_G$ (nm)</td>
<td>20</td>
</tr>
<tr>
<td>Fin thickness, $T_{SI}$ (nm)</td>
<td>10</td>
</tr>
<tr>
<td>Oxide thickness, $T_{OX}$ (nm)</td>
<td>1</td>
</tr>
<tr>
<td>Body doping, $N_{BODY}$ (cm$^{-3}$)</td>
<td>$10^{16}$</td>
</tr>
<tr>
<td>Source/drain doping, $N_{SD}$ (cm$^{-3}$)</td>
<td>$10^{20}$</td>
</tr>
<tr>
<td>Front/back gate thickness, $H_{GF}, H_{GB}$ (nm)</td>
<td>34</td>
</tr>
<tr>
<td>Front/back gate spacer thickness, $L_{SPF}, L_{SPB}$ (nm)</td>
<td>12.5</td>
</tr>
<tr>
<td>Underlap near source/drain, $L_{UN}$ (nm)</td>
<td>5</td>
</tr>
<tr>
<td>Fin pitch, $F_P$ (nm)</td>
<td>60</td>
</tr>
<tr>
<td>Supply voltage, $V_{DD}$ (V)</td>
<td>0.9</td>
</tr>
<tr>
<td>Workfunctions (SG &amp; IG mode), $\Phi_{GN}/\Phi_{GP}$ (eV)</td>
<td>n/p-type: 4.4/4.8</td>
</tr>
<tr>
<td>Workfunctions (ASG mode), $\Phi_{GF}/\Phi_{GB}$ (eV)</td>
<td>n-type: 4.4/4.8; p-type: 4.8/4.4</td>
</tr>
</tbody>
</table>

FinFETs. However, different dopant implantations in the front and back gates lead to extra cost and yield concerns. The gate width is quantized by the number of fins, and is obtained by multiplying the fin count with twice the fin height ($H_{FIN}$) for SG/ASG-mode FinFETs and just the fin height for IG-mode FinFETs.

The key transistor-level design parameters of the 22nm FinFETs considered in this dissertation are summarized in Table 1.1 [73, 74]. The values of $L_G$, oxide thickness ($T_{OX}$), and $T_{SI}$ are obtained from [22]. The values of $H_{FIN}$, body doping ($N_{BODY}$), source/drain doping ($N_{SD}$), $H_{GF}/H_{GB}$, $L_{SPF}/L_{SPB}$, $L_{UN}$, and fin pitch ($F_P$) are suggested by the 2D TCAD FinFET models presented in [73]. $\Phi_{GN}$ and $\Phi_{GP}$ are the workfunctions for symmetric n-type and p-type FinFETs (SG and IG modes). $\Phi_{GF}$ and $\Phi_{GB}$ refer to the workfunctions of the front and back gates of the ASG-mode FinFETs, respectively. These 2D TCAD FinFET models capture the simulation accuracy of 3D TCAD models (error less than 3%) at several orders of magnitude less CPU time (200× faster in total simulation time). Based on the 2D models, Synopsys Sentaurus TCAD [75] is used to calibrate circuit-level parameters of logic gates and memory cells. These results are then included in our simulation framework FinCANON [76].
1.5 PVT Variations

Even though FinFETs exhibit significant advantages, PVT variations are still one of the biggest challenges faced by circuit designers [77]. Process variations in FinFETs are mainly caused by lithographic constraints [77] and difficulties in gate workfunction engineering [78], which are manifested as large spreads in leakage and delay. The impact of process variations is especially severe on caches, as memory cells are usually minimum-sized to maximize density [79]. Such variations may result in several adverse effects on memory cells, such as large deviations in access latency, leakage power difference between cache banks, and unstable read/write operations. At the system level, process variations are classified as systematic (e.g., lithography exposure and defocus aberrations) and random (e.g., line edge roughness). Die-to-die (D2D) and wafer-to-wafer (W2W) variations, which result mainly from systematic components, refer to parametric variations that contribute equally across a die. Within-die (WID) variations, which are caused by both systematic and random components, correspond to variability within a single die and are manifested as spatial correlations among transistors.

Supply voltage variations are usually observed in large integrated circuits (ICs) due to time-varying voltage drops in the power supply network [20]. These variations are mainly caused by IR-drop of the power grid and \( \frac{di}{dt} \) events [80]. IR-drop is determined by the effective path resistance between the voltage regulator module (VRM) and the circuit components. Due to the long distribution paths, it is possible for designs to deliver insufficient voltage to some system components. On the other hand, \( \frac{di}{dt} \) events are caused by changes in operating currents over time. Supply voltage variations are manifested as spikes and drops on the operating supply voltage rails and may adversely affect system performance. They can be mitigated by power/grid planning tools. For high-performance and low-voltage CMP designs,
it is becoming critically important to incorporate voltage variation effects into the fault-tolerant margins.

Thermal variations and hotspots are becoming major causes for electronic failures [81, 82, 83, 84, 85]. Researchers have shown that temperature varies significantly across a CMP. For the Alpha processor and several modern many-core designs, the temperature difference between the hot spots and inactive regions can be as high as 30°C [86]. Thermal variations lead to significant timing uncertainty, prompting wider timing margins, and poorer performance. Approximately 5~6% variation in timing is observed for every 20°C change [87]. The delay variation is mainly caused by thermally-induced variations in the input capacitance \( C_{in} \), output capacitance \( C_{out} \), and output resistance \( R_{out} \) of logic gates. The variation in \( R_{out} \) of a logic gate can be as much as 16.5% from 25°C to 125°C, on an average [88]. Leakage power grows exponentially as the operating temperature increases. It is reported in [89] that at 105°C, the leakage power of a normally-biased \( (V_{HI} = 1.0V \text{ and } V_{LOW} = 0V) \) router can be as high as 7× as compared to a reverse-biased \( (V_{HI} = 1.2V \text{ and } V_{LOW} = -0.2V) \) router. Researchers have also pointed out that thermal variations in modern many-core processors are becoming significant and can no longer be neglected [90].

Over the past few years, the impact of PVT variations on CMPs has attracted attention. Delay variation and the probability of failure in NoC links are analyzed in [91]. Delay variation effects on router components are studied in [92]. In [93], the authors simultaneously consider the effects of delay variations on both routers and links. The impact of process and temperature variations on power and energy-delay-product-per-flit metrics for different NoC architectures is investigated in [94]. Several approaches have been proposed to mitigate the impact of PVT variations on CMPs using dynamic voltage scaling (DVS) [95], dynamic frequency scaling (DFS) [17], and body biasing. The globally asynchronous, locally synchronous (GALS) concept is adopted in some CMP designs in which individual cores and the network fabric
operate at their own maximum frequencies \cite{93}. Unfortunately, most of the previous studies model PVT variations in terms of the transistor $V_{th}$ only, failing to incorporate the impact of process parameters. FinFET researchers have extensively investigated the impact of D2D and WID variations on device-level and circuit-level performance \cite{72, 77, 78, 96}. Modeling methodologies for characterizing the effects of FinFET process variations have been proposed \cite{77, 97}. Issues dealing with mitigating their impact have also attracted attention \cite{63}. However, the impact of PVT variations on FinFET-based cache and NoC designs is still not well understood. A thorough understanding of manifestation of such variations is essential and their impact on performance and power consumption has to be studied in order to design better CMP systems that are robust to these variations.

1.6 Dissertation Contributions

The major contributions of this dissertation are the methodologies of estimating power, timing, and PVT variations of FinFET-based architectures and an integrated simulation framework called FinCANON, which allows rapid estimation of power-performance trade-offs for FinFET-based caches and on-chip networks at an early design stage by varying the cache and network microarchitecture in a parameterized fashion. Specifically, FinCANON includes a FinFET design library, CACTI-PVT, and ORION-PVT. The FinFET design library provides circuit-level parameters, such as capacitance, resistance, leakage, as well as PVT variation trends of FinFET logic gates and memory cells. Based on a statistical static timing analysis (SSTA) \cite{98} technique and macromodel-based methodology, we have developed PVT variation models for delay and leakage using Yang’s gate-level macromodels \cite{88}, taking into account spatial correlations, to characterize the impact of PVT variations on FinFET-based caches and NoCs. FinCANON is built on top of CACTI-PVT and ORION-PVT,
which are enhanced from CACTI-FinFET \cite{99} and ORION-FinFET \cite{89} infrastructures for modeling FinFET-based caches and NoCs, respectively. CACTI-FinFET and ORION-FinFET are enhanced from CACTI \cite{100} and ORION \cite{101}, respectively. Based on the FinCANON framework, we also present a new flow control mechanism for FinFET-based NoCs, called variable-pipeline-stage router (VPSR), for fine-grain DPM. Finally, we present a flow control mechanism for on-chip networks, called enhanced token flow control (ETFC), which improves upon the previous TFC mechanism through a combination of better buffer utilization and guaranteed express virtual channels (GEVCs). We describe these contributions in more detail next.

### 1.6.1 FinFET design library

The FinFET design library contains circuit-level characteristics as well as their variation trends with respect to various PVT parameters for FinFET logic gates and memory cells. Using quasi-Monte Carlo (QMC) device simulations \cite{102}, we identify the major sources of process variations in FinFETs, and quantify their impact on delay and leakage spreads in FinFET logic gates and memory cells. We incorporate the above information, along with the characteristics of FinFET logic gates and memory cells, into the cell-based FinFET design library. The library also incorporates voltage generators (VGs) to enable the modeling of back-gate biasing (BGB). BGB enables fine-grained control of the back-gate bias voltages, such that $V_{th}$ of the FinFET cache or NoC can be easily adjusted. Note that BGB should not be confused with body biasing (BB) used in CMOS. BGB has a much better control of $V_{th}$ than BB. We further derive two variation models to characterize the impact of PVT variations. The first delay variation model is based on an SSTA \cite{98} technique whereas the second leakage variation model is inherited from a macromodel-based methodology \cite{77}. This FinFET design library can be employed in various architectural simulation tools. It has the following features:
• It is based on 2D TCAD FinFET models that capture the simulation accuracy of 3D TCAD models.

• It supports all three types of FinFET design styles (SG, IG, and ASG modes).

• It supports delay and power modeling for FinFET logic gates and memory cells at the 22nm node and beyond.

• The PVT variation models can be applied to various circuits and CMP components.

• The FinFET design library is modular. New logic gates and memory cells can be easily incorporated into the FinFET design library.

1.6.2 FinCANON

FinCANON [76] is a design and simulation framework for fast and accurate prediction of delay and power for FinFET-based cache and NoC components, under PVT variations, considering the effect of spatial correlations. FinCANON is built on top of CACTI-PVT and ORION-PVT, which are enhanced from CACTI-FinFET [99] and ORION-FinFET [89] infrastructures for modeling of caches and on-chip networks, respectively. FinCANON incorporates the FinFET design library. Unlike prior studies, all kinds of process, supply voltage, and temperature variations are taken into account together. We consider variations in router components as well as interconnects. Additionally, we model cache banks with various logic and memory design styles, e.g., 4T, 6T, and 8T SRAM cells. Moreover, the cache and NoC models in FinCANON have been significantly updated to be more modular and scalable. We have used FinCANON to run a large number of simulations, varying the values of $L_G$, $T_{SI}$, $T_{OX}$, $\Phi_{GN}$, and $\Phi_{GP}$. We have rigorously investigated cache and NoC delay and leakage for various FinFET design styles, supply voltages, temperatures, cache capacities, and
memory cell design styles. We have experimented with the Princeton Application
Repository for Shared-Memory Computers (PARSEC) real-traffic benchmarks [103].
It is a benchmark suite composed of multithreaded programs. It is specifically de-
signed to represent emerging workloads of next-generation shared-memory programs
running on CMPs. The contribution of this work are as follows:

- FinCANON enables architects to evaluate the impact of PVT variations on
caches and on-chip networks at an early design stage, and quantify tradeoffs
among different architectures.

- CACTI-PVT is the first simulation framework for modeling FinFET-based
caches. In addition to the features provided in the original CACTI, CACTI-
PVT allows various combination of FinFET logic gates and memory cell styles
to be included in a cache.

- ORION-PVT is the first simulation framework for modeling FinFET-based
NoCs. It models not only the power consumption in an NoC, but also the
delay and area of router components. ORION-PVT adopts CACTI-PVT for
modeling the input buffers inside a router, and thus can accurately capture the
power and delay of those buffers.

- Both CACTI-PVT and ORION-PVT are based on the FinFET design library.
Therefore, they offer accurate modeling of FinFET-based caches and NoCs at
the 22nm node.

1.6.3 Variable-pipeline-stage router

We present a detailed design of VPSR [70, 71]. VPSR allows the number of router
pipeline stages to be varied in response to a network’s traffic requirement. It is espe-
cially well-suited to FinFET-based interconnection networks, with the aim of dynam-
ically managing the power consumption while enhancing throughput. While popular
techniques like DVS or DFS can reduce active-mode power, they incur significant transition latency and energy overheads. Adaptive back-gate biasing (ABGB) [70], on the other hand, can switch the back-gate bias voltages of FinFETs very quickly, with little transition energy overhead. ABGB dynamically controls the back-gate bias voltages of FinFETs in order to produce various power and delay characteristics. ABGB on FinFET-based caches has been explored in [104]. We incorporate voltage generators into VPSR to enable run-time exploitation of the ABGB concept. The features of VPSR include:

- It separates input buffers of a router port into normal and slow banks. Normal banks are normally-biased to improve the throughput of a router, while slow banks are reverse-biased to save leakage power.

- A flow control algorithm is used in VPSR to dynamically adjust the ratio of normal banks to total banks based on the incoming traffic load of a router port.

- Each router port can switch among three types of pipelines to achieve significant runtime power reduction while delivering energy-delay-throughput improvement in the context of FinFET-based interconnection networks.

### 1.6.4 Enhanced token flow control

We have developed a flow control mechanism, called ETFC, to provide better network traffic regulation than previous flow control techniques. ETFC has been enhanced from TFC [23], which aims to approach the communication energy-delay-throughput of dedicated wires. We analyze the token passing protocol in TFC and propose a revised protocol to improve buffer utilization and enhance the use of tokens. ETFC incorporates a new concept involving the use of guaranteed tokens to build GEVCs. Instead of enabling the bypassing of unrelated flits, a GEVC allows the bypassing of all the flits belonging to a particular packet along the token route to their destination.
After the tail flit traverses the route, GEVC expires itself. A brief summary of the contributions of this work are as follows:

- Normal tokens are forwarded a limited number of hops, which improves efficiency of router resource usage.

- A guaranteed token provides a guarantee to a single packet for resource availability at the endpoint node of the token route.

- GEVCs are used to detour packets from congested regions.

1.7 Dissertation Structure

The remainder of this dissertation is organized as follows. Chapter 2 introduces past work related to this dissertation. Chapter 3 presents the FinFET design library and PVT variation models. Chapter 4 presents the FinCANON framework. Chapter 5 presents the detailed design and evaluation of VPSR. Finally, Chapter 6 provides a summary of the dissertation and points to future research directions.
Chapter 2

Related Work

In this chapter, we review a wide range of related work from the literature. In Sections 2.1 through 2.3, we introduce a series of techniques for modeling caches, NoCs, and PEs. These techniques model different architectural design metrics, including power, delay, and area, for different types of systems. In Section 2.4, we survey different simulation frameworks for CMP performance simulation. These frameworks generate run-time statistics of CMPs with different levels of details. These statistics are used to estimate, optimize, and manage power consumption and timing of a CMP system. In Section 2.5, we survey past work on PVT variation models, which are relevant to the PVT models developed in the next chapter. In Section 2.6, we survey relevant techniques for NoC flow control for implementing high throughput and low power on-chip networks.

2.1 Cache Modeling

CACTI [100] is a widely-used simulation tool from HP Labs that models power, area, and access delay of RAM-based structures in a hierarchical fashion. Most of the recent cache models [33, 105, 106, 107, 108, 109] in the literature are similar to, or derived from, CACTI’s infrastructure. The recent versions of CACTI allow modeling of
Figure 2.1: Cache structure in CACTI [100].

DRAMs [108], NUCAs [33], and 3D die-stacked off-chip DRAM [109]. Fig. 2.1 shows the cache structure modeled in CACTI. It accepts as inputs various cache parameters, such as capacity, associativity, block width, number of ports, number of banks, technology node, output width, tag width, access mode, memory type, temperature, device type, peripheral circuitry, interconnect technology, wire type, H-tree details, page size, level of prefetch, and burst length. CACTI divides a cache into eight major components: decoders, word lines, bit lines, sense amplifiers, comparators, multiplexers, output drivers and inter-bank wires. It also divides the cache into the tag and data sub-arrays. Depending on the cache configurations, CACTI calculates the required transistor sizes inside each cache component. For instance, the wordline driver size in a cache is determined by the dimension of the subarray. CACTI uses the logical effort method to optimize transistor sizes and delay such that the timing constraints are met. This methodology is also applied to other parts of the cache components, such as decoders, H-tree drivers, etc. CACTI allows designers to optimize the dynamic energy per access, leakage power, or access time. An optimizer is invoked in parallel to determine the best cache configuration under area, access time, and
power constraints. CACTI models CMOS-based caches using a CMOS technology library. The technology library contains device-level parameters, such as $L_G$, $T_{OX}$, $V_{th}$, etc., for various technology nodes. Circuit-level parameters, such as capacitances and resistances of logic gates and memory cells, are derived from CACTI's analytical models and device-level parameters in its technology library. The delay and power of CMOS-based caches are then derived from the circuit-level parameters of logic gates and memory cells. However, as the feature size shrinks to the deep submicron regime, the analytical models become increasingly complex. Many more device parameters are required for accurate gate- and circuit-level modeling. For FinFETs in the deep submicron regime, satisfactory analytical models are still not available [97]. CACTI also limits its memory cell styles to 6T SRAM and 1T1C DRAM cells. In the past decade, CACTI has been incorporated into several other tools [33, 34, 110].

2.2 On-chip Network Modeling

A number of NoC models [34, 101, 111, 112, 113, 114, 115, 116] have been proposed to estimate the power consumption in an on-chip network. Initial work in this area considers power dissipated in switches and links [111]. The power consumed by a crossbar switch is obtained based on the transistor count. This work presents detailed models for short-circuit power and link delays, however, ignores the leakage power consumed in a router. In [112], a methodology is introduced to automatically build the energy model of an NoC. It is based on an architectural regression model for each router node. However, it fails to model router components in detail. In [113, 114], an analytical framework is proposed to estimate network resource utilization for a number of architectures to calculate network delay and power. In [115], a power estimation model based on logic synthesis is presented. This work develops a register-transfer level (RTL) description of an NoC router, and measures switching activities.
of the router under various traffic scenarios. However, it cannot be easily mapped to
different router configurations.

ORION [34] is a CMOS-based power-performance simulator for on-chip networks. It estimates router power consumption using built-in dynamic and leakage power models for various types of buffers, crossbars, and arbiters [34, 116]. Its models include capacitance equations [34] and leakage current tables [116] for various router components. The capacitance equations formulate the load capacitance at each node of the circuit for calculating the dynamic power. The leakage current table specifies the leakage current per transistor width over length. ORION is developed on top of an older version of CACTI. As a result, its transistor models come from CACTI’s CMOS technology library. ORION inherits the analytical models from CACTI, and uses them to calculate delay and power of the components in on-chip routers. ORION’s models have been validated against the MIT RAW processor [1], with power estimation error less than 10%. However, ORION does not incorporate clock and link power in its simulations. In [5, 35], it is estimated that when operated at 4GHz, clock power constitutes about 11% of the tile power. The peak power from routers and links is about 28% of the tile power [5, 35]. Therefore, they cannot be ignored. In addition, ORION only estimates leakage power at a fixed temperature (80°C) [116].

ORION 2.0 [101] is a successor to ORION, which incorporates new power models for clocks, links, and D flip-flops (DFFs). The power models of VC allocators and arbiters have also been updated. ORION 2.0 extends scaling factors in ORION’s technology file, such that it is able to model technologies down to the 32nm node. It also contains a new area model to estimate the size of router components.

Fig. 2.2 compares the router power breakdown reported by ORION for three different technology nodes. The operating frequency is 1GHz. It can be seen that leakage power grows significantly as technology scales down. At the 50nm node, leakage power is 2.8× relative to the 100nm node. This is mainly attributable to SCEs, including
Figure 2.2: Router power breakdown for bulk CMOS.

DIBL, threshold voltage roll-off, and sub-threshold slope degradation. ORION calculates the circuit-level parameters of router components at these technology nodes by scaling from the 100nm node, rather than deriving them from exact transistor sizes. In Chapter 4, we present ORION-PVT in which FinFETs as well as logic gates are carefully calibrated at each technology node, and not simply scaled.

2.3 Processor Modeling

In the past decade, several processor modeling tools have been proposed [33, 110, 117, 118, 119, 120]. SimplePower [117] is the first cycle-accurate framework for runtime processor power estimation. It is based on the conventional five-stage pipeline processor architecture. At each clock cycle, SimplePower calls its RTL power estimation interfaces for all activated functional units. In [118], a power table is developed for each functional unit inside a processor, which includes the power consumption values.
for various input vectors. If the technology or functional units are changed, the power tables in these interfaces are changed as well.

Wattch [110] is a widely-used architecture-level power model for microprocessors. It has been embedded in various performance simulators (e.g., SimpleScalar [121], GEMS [122], etc.). It uses detailed circuit-level structures of various processor components (e.g., array structures, combinational logic, etc.), processor microarchitecture configurations, technology parameters, and runtime activity counts of the processor components as inputs for calculating power. It models out-of-order processors based on the synthetic reservation update unit (RUU) model of SimpleScalar [33, 121]. The runtime activity counts are obtained from the performance simulator used. These runtime activities are then fed into Wattch’s power model to calculate the dynamic power. Wattch has several drawbacks. First, Wattch models dynamic power only, and does not have a well-defined model for leakage power. A leakage model called HotLeakage [123] was added to Wattch later as an extension. However, it only provides a simple transistor-level leakage power model, and does not model leakage power at the component level (e.g., functional units, caches, etc.) in detail. Second, Wattch uses an older version of CACTI as its cache model, which does not model cache components in detail. Third, Wattch uses scaling factors based on the 0.8µm technology node. These factors are inaccurate for making predictions in the deep-submicron regime. When the technology scales down to the 22nm node and beyond, the scaling methodology can no longer capture the behavior of transistors correctly. Fourth, Wattch models power consumption without considering timing and area. Last, Wattch does not support CMP architectures, but only considers uniprocessor architectures. In [119], the authors extend Wattch’s power model with the use of hardware performance counters to measure the runtime power consumption of a microprocessor.

HotSpot [124] is an architectural thermal model for microprocessors. It constructs a multi-layer lumped thermal RC network to model the heat dissipation path from the
silicon die through the cooling package to the ambient. In HotSpot, the silicon die is partitioned into functional blocks based on the floorplan of the microprocessor, with a thermal RC network connecting the various blocks. HotSpot can be readily used to model on-chip network routers. Each router within the chip floorplan is modeled as a block, and a thermal RC network is constructed in the same fashion as when the blocks were functional units within a microprocessor.

McPAT (Multicore Power, Area, and Timing) [33] has been developed by HP Labs and is an integrated power, area, and timing modeling framework for CMP architectures. It is based on CACTI 6.5’s infrastructure, which has a more detailed cache model and enables modeling of NUCAs. It models processor cores in a hierarchical, fine-grained fashion, including models for the components of a complete CMP, e.g., in-order and out-of-order processor cores, NoCs, shared caches, and integrated memory controllers. All the registers, caches, and content-addressable memories (CAMs) inside a processor core are modeled by CACTI’s memory model. The power and delay of ALUs and FPUs are scaled from those of existing processors. Fig. 2.3 shows the modeling hierarchy of McPAT. McPAT uses an extensible markup language (XML) interface to communicate with performance simulators. There are several drawbacks of McPAT as well. First, it has not been integrated into any performance simulator available outside HP. Second, it only has a simple router model, and no detailed NoC model. Third, it also uses the scaling technique to model different technologies, which is not very accurate in the deep-submicron regime. Fourth, it does not have a detailed model for ALUs and FPUs. Scaling the power and area numbers of ALUs and FPUs from those of existing processors does not guarantee correctness for different processor configurations and technology nodes. Last, it is not able to model FinFETs due to a lack of analytical models.
2.4 CMP Performance Simulation

GEM5 [125] is a full-system simulation framework that originates from the GEMS [122] simulation framework. GEM5 supports a wide range of simulation capabilities, including various types of instruction set architectures (ISAs), CPU models, and memory coherence protocols. The ISAs supported by GEM5 include Alpha, ARM, MIPS, Power, SPARC, and X86 [125]. The CPU models supported by it include non-pipelined single-cycle CPU models, pipelined in-order model, and pipelined out-of-order model. The memory models supported by it include a classic model and Ruby model. The classic model allows fast simulations, but does not model the behavior of the memory system. On the other hand, Ruby enables detailed modeling of a wide variety of cache-coherent memory systems.

GARNET [126] is a cycle-accurate performance simulator for interconnection networks that incorporates the original ORION router power model [34]. GARNET is embedded in the GEMS full-system simulator, which provides a framework for modeling the runtime statistics of CMP systems. GEMS itself has a very simple network model, which is included in Ruby. Unlike the simple network model, GARNET models the router microarchitecture in detail, including all relevant resource contentions and flow control timing. It models on-chip networks in an event-driven fashion. As
long as a flit is passed between the routers or moved inside the router pipeline, the event is added to an event queue, which is examined in each cycle. GARNET enables modeling of fixed five-stage pipeline baseline routers, as well as flexible pipeline routers with buffers inserted. It allows various network configurations. It accesses Ruby’s configuration file and determines the router and network configurations to be used. It can model synthetic traffic patterns, which include deterministic, random, bit-reverse, and tornado traffics. It also works together with GEMS to provide benchmark modeling, e.g., SPEC [127], SPLASH-2 [128], and PARSEC [103]. The routing algorithms can be configured by the user as well. Each router in GARNET has a routing table that is populated at configuration time. The routing table is examined in the RC stage and updated to determine the output port to which a flit is to be forwarded. GARNET allows the user to add flow control algorithms to it. Various flow control algorithms, such as EVC, TFC, and ETFC, have been incorporated into the GARNET framework. As a result, GARNET provides a complete simulation platform to simulate the power consumption based on the traffic in the network.

2.5 PVT Variation Modeling

Under PVT variations, delay and power of a circuit are random variables due to variations in device parameters. Traditionally, static timing analysis (STA) has been an indispensable step for delay verification of an IC. It takes nominal values of electrical parameters as inputs, and has a linear runtime with respect to the size of the circuit. STA is used to evaluate the impact of variability in delay using so-called corner-case analysis over the entire parameter variation space, which requires a huge amount of computation time and resources to achieve accurate modeling, and thus has been shown to be impractical [98, 129, 130, 131, 132, 133, 134, 135, 136, 137]. WID variations often exhibit spatial correlations, where devices that are close to
each other have a higher probability to be strongly correlated than devices that are
spatially far from each other. WID variations can occur due to topological dependencies of device processing, such as chemical mechanical polishing (CMP) effects and optical proximity effects [137]. This issue has been the subject of a number of recent works [137, 138, 139, 140, 141, 142]. These spatial variations therefore posed a challenge for STA, giving rise to the need for SSTA [98] approaches.

SSTA offers an alternative approach that takes distributions of electrical parameters as inputs and treats delays of logic gates in a circuit as distributions. The delays characterized by SSTA are modeled as Gaussian functions that are dependent on process parameter variations. These functions are then approximated using first-order Taylor-series expansion that takes into account both correlated and independent randomness arising from different sources of variation [98, 136]. SSTA falls into two categories: path-based algorithm [98] and block-based algorithm [136]. Given a timing graph constructed from a circuit, the former algorithm selects a set of paths and traverses the paths in a depth-first manner, while the latter traverses the timing graph in a levelized breadth-first manner. In addition to the linear models, alternative SSTA delay models based on quadratic polynomial regression have been presented [138, 143, 144, 145], which are able to capture nonlinear dependencies of gate delay on process parameter variations. A multivariate regression based SSTA technique is proposed in [146] to further consider the effects of spatio-temporal variations. It has been evaluated on a simple 32KB cache using CACTI 1.0 and shown that less than 5% error is achieved as compared to SPICE simulation.

Several techniques have been proposed to model spatial correlations in a circuit. To account for WID variations, a multi-grid based approach has been proposed that divides a die into grids [137]. In such an approach, a PVT parameter of all the transistors within the same grid is modeled by a single random variable (RV), thus avoiding the need to consider parameter correlations between individual transistors, which re-
quires taking millions of transistors into account and is computationally prohibitive. The correlated RVs among different grids are translated using principal component analysis (PCA) to a different coordinate system such that all translated RVs are independent of each other [98]. A technique based on Karhunen-Loève expansion (KLE) is proposed [139] to reduce the number of RVs in the multi-grid based approaches while maintaining accuracy. This work models the parameters of the device as stochastic processes over the spatial domain of a die, and provides a mechanism to easily calculate the correlations without the need of a covariance matrix. Researchers have also proposed a method [142] to extract spatial correlations from real chip measurements. This work also extends the multi-grid based approach from conventional uniform grids to non-uniform grids based on the assumption that process control at the center area of a chip is better than at the boundaries. Therefore, central grids are larger than those located at the boundaries.

Most of the previous work on statistical performance analysis has focused on SSTA, and only a few works have explored the variation of leakage power under PVT variations [147, 148, 149, 150, 151, 152, 153, 154, 155]. In [147], analytical methods are presented to estimate the statistical mean and variance of leakage current in a single inverter. In [148], the authors further extend their analytical models to the circuit level. In [149], a method is proposed for analyzing the leakage current distribution of a chip under process variations, considering the spatial correlations among parameters. This work models leakage current of a logic gate by an empirical curve-fitting model, called the leakage macromodel [77]. In such a model, the leakage current of a logic gate is approximated by a lognormal distribution. In [150], KLE is used to facilitate the calculation of covariance between logic gates in a circuit. In [151], an analytical model is presented for single-device leakage distribution in FinFET technology, considering variations in \( L_G \) and \( T_{SI} \). A macromodel technique, which statistically evaluates leakage current in FinFET-based circuit, has been proposed recently in [77].
2.6 NoC Flow Control

EVC and TFC both utilize the concept of router pipeline bypassing to achieve shorter network latency and reduce power consumption. Lookaheads (LAs) are used to set up the resources in the intermediate routers along the packet route, as described in the previous chapter.

2.6.1 Credit-based flow control mechanism

The basic credit-based flow control mechanism [12] is based on the credits stored in the router port. The number of credits indicates the number of available buffers in the downstream router. A flit is sent to the downstream router only when it obtains a credit. If the credit is equal to zero, no flit is allowed to be sent to the downstream router. When a buffer is freed in the downstream router, an acknowledge signal is sent back, and the credit count is incremented. This mechanism controls the packet flow by back-pressuring (if no credit is available), such that flit loss is avoided.

2.6.2 Express virtual channel

EVC [24] is a flow control mechanism that allows packets to virtually bypass intermediate routers along their path in a completely non-speculative fashion, thereby lowering the energy/delay towards that of a dedicated wire while simultaneously approaching ideal throughput with a practical design suitable for on-chip networks. Flits that acquire a \( k \)-hop EVC can bypass \((k - 1)\) router hops. EVCs can be either static or dynamic. The former provides a fixed uniform EVC length to connect source and sink nodes throughout the network. The latter enables every node to become source or sink, and allows EVCs of varying length. Dynamic EVCs provide routing flexibility, but require more complex protocols for signaling and exchanging of control information as well as hardware overhead. EVCs achieve up to 84% reduction
in packet latency and up to 23% improvement in throughput. They reduce router energy consumption by up to 38% over a state-of-the-art packet-switched network design. They, however, have several limitations. First, the virtual path created by an EVC allows packet bypassing within a single dimension only. Hence, multiple EVCs are required when the packet route involves turns. In addition, the maximum length of an EVC is limited by the amount of buffering available at router ports [23]. Moreover, buffers and VCs are statically allocated to EVCs, which prevents dynamic sharing of available resources. An enhanced version of EVC utilizing global-line interconnects is proposed in [156], which shows a 7.6% improvement in network latency over the original EVC design.

### 2.6.3 Token flow control

TFC [23] is a flow control mechanism that attempts to close the performance and energy gaps between the packet-switched network and the ideal interconnection fabric (direct connections) through the use of tokens. Tokens are sent by each router to the neighborhood defined by a fixed number of hops to indicate resource availability at the router input ports. The forwarding routes are called token routes. A packet possessing a token that matches its route may bypass the intermediate nodes along the token route and travel directly to the destination node. Two types of tokens are provided: normal and guaranteed. A normal token is broadcast up to $d_{\text{max}}$ hops away and provides a hint for VC and buffer availability at an input port of the source router. On the other hand, a guaranteed token is unicast $g_{\text{max}}$ hops away and guarantees VC and buffer availability at the source router. A token is turned off by the source router if either the number of free buffers or the number of free VCs drops below a threshold. The VC and buffer thresholds for guaranteed tokens, $v_{g_{\text{th}}}$ and $b_{g_{\text{th}}}$, are higher than those for normal tokens (denoted as $v_{\text{th}}$ and $b_{\text{th}}$). Propagation of tokens between adjacent nodes is done through separate point-to-point wires, which are much
narrower than the data channels and, hence, require minimal overhead. When one token route ends, another token route can be chained to it to enable bypassing over multiple token routes. Conflicts between multiple flits from different input directions using tokens for the same output port are resolved by a switch port priority vector. This vector has one entry for each output port and denotes the input port that has priority to access the output port. The switch port priority vector can be updated dynamically. Flits using guaranteed tokens are given higher priority than those using normal tokens. If an LA carrying a token is killed due to conflicts with other LAs, its corresponding data flit is buffered and has to go through the normal pipeline. TFC has been demonstrated to reduce packet latency by up to 77.1% and average router energy consumption by up to 39.6% as compared to a state-of-the-art baseline packet-switched network.
Chapter 3

FinFET Design Library

3.1 Introduction

The FinFET design library contains a FinFET logic gate library, memory library, and FinFET model cards for various technology nodes. The FinFET model card describes FinFET characteristics at the device level. It contains the FinFET design parameters listed in Table 1.1. Based on the specified technology node, the FinFET design library chooses the corresponding model card, searches the associated libraries, and provides the capacitance, resistance, leakage, area, and delay characteristics of the FinFET components. The FinFET logic gate library contains characteristics of INV/NAND/NOR gates for various logic styles (SG, IG, and ASG modes), gate sizes, supply voltages, and temperatures. The layouts of the 22nm FinFET standard logic cells are available from [72]. The FinFET memory library contains characteristics of several FinFET SRAM cell styles. The FinFET design library also incorporates the power, leakage, voltage transition time, area, and other characteristics of voltage generators. Voltage generators are used to generate the bias voltages for the back gates of FinFETs, and are explained in detail in Section 3.3. Instead of extending the analytical device models in the original CACTI to FinFETs, which are still under
Figure 3.1: FinFET design library construction flow

study [97], FinCANON adopts a cell-based methodology in the FinFET design library similar to the lookup-table (LUT) approaches used in [97]. This LUT-style modeling approach enables fast and effective modeling of PVT variations in FinFETs.

### 3.1.1 FinFET design library construction

Fig. 3.1 shows the FinFET design library construction flow. It involves the use of the Sentaurus TCAD structure editor and device simulator. The TCAD structure editor, together with the custom parameter file, generates the 2D FinFET structure files for both nFinFETs and pFinFETs. To generate the FinFET design library, we choose the PVT parameters to be simulated and specify the logic configurations. We then use the
TCAD logic file generator to generate the command files for simulating various types of logic gates and memory cells with the specified PVT parameter values. TCAD simulates the generated command files based on the 2D FinFET structure files. A parameter extractor processes the output files and extracts the circuit-level parameter values required for the delay and leakage variation models. The FinFET design library stores the extracted parameter values. When more simulations are necessary (e.g., for different bias voltages), the simulation flow returns to the first step. The procedure is repeated until all the simulations are done.

Fig. 3.2 shows the block diagram of the parameter extractor in Fig. 3.1. Several shell scripts are used first to process the TCAD output files and extract values of circuit-level parameters, such as $C_{in}$, $C_{out}$, $R_{out}$, and leakage current ($I_{Leak}$). These parameter values are then inserted into the MySQL databases. Finally, MATLAB is used to extract values of the sensitivity parameters for the delay and leakage variation models.
3.2 PVT Variation Models

In this section, we present the PVT variation models employed in FinCANON. It includes delay and leakage variation models. Both models take technology and temperature into account. The variation trends in circuit-level parameter values for various temperatures and FinFET logic design styles are calibrated, and incorporated into the FinFET design library. These trends mainly describe how the circuit-level parameters change as the PVT parameters vary from their nominal values. The extent of systematic and random variations of the PVT parameter values can be defined by the user, or can be imported from real fabrication data. We assume that the process parameters and supply voltage (PV) variations are Gaussian RVs [98, 137], with means equal to their nominal values, and 3σ/µ equal to 10%. The delay is normally distributed, while the leakage power is lognormally distributed. The temperature variation model is based on the LUT methodology, which is described later. The outputs of the PVT variation model are the nominal values and standard deviations (STDs) of the delay and leakage power of the cache and NoC components.

The rectangular grid-based method [137] is used to model the spatial correlation of the PV parameters. It hierarchically partitions a die into multiple levels of grids. The grid size is determined by the floorplan of the chip. For regular structures, it is commonly adjusted to the size of basic components [137]. For instance, mats are assigned to different grids in a cache bank. For CMPs, each tile is a grid. The variation of a PV parameter p in each grid is modeled by a zero-mean RV Δp. The RVs in different grids or levels are independent. This method can be easily adapted to caches and NoCs, which are inherently organized in a hierarchical grid structure, as shown in Fig. 3.3(a). From the top to the bottom, the levels correspond to the core, bank, and subarray. Note that each level can be partitioned into any number of grids, not necessarily just four. The variation of p in a particular subarray is represented by the Δp’s of the grids that cover it. For example, the TSI variation in subarray S0
can be represented as:

\[ T_{SI,S0} = T_{SI,nom} + \Delta T_{SI,S0} + \Delta T_{SI,B0} + \Delta T_{SI,C0} \quad (3.1) \]

where \( T_{SI,nom} \) is the nominal value of \( T_{SI} \), and \( \Delta T_{SI,S0} \), \( \Delta T_{SI,B0} \), and \( \Delta T_{SI,C0} \) are zero-mean normal RVs representing the \( T_{SI} \) variation in grid \( S0 \), \( B0 \), and \( C0 \), respectively. The variance of \( T_{SI,S0} \) is, therefore, expressed as:

\[ Var(T_{SI,S0}) = Var(\Delta T_{SI,S0}) + Var(\Delta T_{SI,B0}) + Var(\Delta T_{SI,C0}) \quad (3.2) \]
We assume that correlation exists only among the same type of parameters, and there is no correlation between different types of parameters \[77\]. The covariance of \( p \)'s of different grids is represented by the sum of variances of \( \Delta p \)'s of their common parent grids. For example, the covariance of \( T_{SI,S_0} \) and \( T_{SI,S_5} \) is written as:

\[
Cov(T_{SI,S_0}, T_{SI,S_5}) = Var(\Delta T_{SI,B_0}) + Var(\Delta T_{SI,C_0})
\]

(3.3)

As a result, gates that lie within close proximity of each other on a die have a stronger correlation, while gates that lie far apart share few common components, and thus have a weaker correlation.

To deal with the logic gates of links between grids (e.g., the drivers of the address lines between subarrays), we use a binary-tree data structure to model their spatial correlations with the neighboring grids. Fig. 3.3(b) shows a cache bank with several subarrays, and Fig. 3.3(c) the binary tree for the cache bank. In Fig. 3.3(b) \( RV \Delta P_i \) describes the variation of a PV parameter in the corresponding array. These RVs form the leaves of the binary tree in Fig. 3.3(c). \( RV \Delta W_i \) of an intermediate node represents the parameter variation of the links connecting its children nodes. The value of \( \Delta W_i \) is a function of the RVs of its children nodes. Functions \( f_0, f_1, \) and \( f_2 \) depend on the floorplan of arrays as well as the fabrication technology.

### 3.2.1 Delay variation model

The delay variation model translates the PV variations into Gaussian RVs that describe gate delay variations. These RVs are then used for calculating the delay distribution of the target component, the overall component access delay, or for the purpose of optimizations. In most cases, these delays are not linearly dependent on device parameters. Thus, they are actually not Gaussian, even when the PV variations are Gaussian. However, when the PV variations are small, we can still model them as...
Gaussians using first-order Taylor expansion [98]. Assume that for a gate \( k \) in the critical path, its delay \( (d_k) \) and output transition time \( (s_k) \) are expressed as functions \( D_k \) and \( S_k \), respectively:

\[
\{d_k, s_k\} = \{D_k, S_k\}(\vec{R}(\vec{P}_m), \vec{C}(\vec{P}_m), s_{k-1})
\]  \hspace{1cm} (3.4)

where \( m \) is the subarray where gate \( k \) resides, \( s_{k-1} \) the output transition time of the previous gate \( k - 1 \), \( \vec{P}_m \) the vector of PV parameters in subarray \( m \), \( \vec{R} \) the vector of load resistances, and \( \vec{C} \) the vector of load capacitances. \( \vec{R} \) and \( \vec{C} \) are functions of \( \vec{P}_m \), where \( \{L_G, T_{SI}, T_{OX}, \Delta \Phi_{GN}, \Delta \Phi_{GP}, V_{DD}\} \in \vec{P}_m \). Load resistances include resistances from the output of logic gates and wires. We base our empirical model on the modification of the well-known Horowitz delay approximation [157] for FinFETs as follows [88]:

\[
D_k = \tau \cdot \sqrt{\alpha_d + A \cdot \left(\frac{s_{k-1}}{\tau}\right)}
\]  \hspace{1cm} (3.5)

where \( \alpha_d \) is used to replace the original \( (\log[V_{th}]^2) \), \( A = 2 \cdot \beta_d \cdot (1 - V_{sw}) \) for the rising edge, \( A = 2 \cdot \beta_d \cdot V_{sw} \) for the falling edge, \( \tau \) is the time constant of the gate, and \( V_{sw} \) is the switching point (50% of the voltage range) of its input and output signals. The values of \( \alpha_d \) and \( \beta_d \) are extracted from TCAD simulations, and are stored in the FinFET design library. They have different values for different gates and sizes.

Based on the TCAD simulation results, we express \( S_k \) with the following empirical model [88]:

\[
S_k = S_{\text{base}} \cdot \left\{\alpha_s + \left[\ln\left(\frac{s_{k-1}}{S_{\text{base}}} + \beta_s \cdot \gamma_s\right)^2\right]\right\} = S_{\text{base}} \cdot \left\{\alpha_s + X^2\right\}
\]  \hspace{1cm} (3.6)

where \( S_{\text{base}} = \ln 9 \cdot \tau \) is the output slope when \( s_{k-1} = 0 \), and \( \alpha_s, \beta_s, \) and \( \gamma_s \) are empirical parameters derived for each type and size of gate.
Using the first-order Taylor expansion, (3.4) can be rewritten as:

\[ \{d_k, s_k\} = \{d_{k,nom}, s_{k,nom}\} \]

\[ + \sum_{p_m \in P_m} \frac{\partial \{D_k, S_k\}}{\partial p_m} \Delta p_m + \frac{\partial \{D_k, S_k\}}{\partial s_{k-1}} \Delta s_{k-1} \]  

(3.7)

where \(d_{k,nom} (s_{k,nom})\) is the nominal value of \(d_k (s_k)\), \(\Delta p_m\) the zero-mean Gaussian RV representing the deviation in PV parameter \(p_m\), and \(\Delta s_{k-1}\) the deviation in \(s_{k-1}\).

The value of \(\frac{\partial D_k}{\partial p_m} (\frac{\partial S_k}{\partial p_m})\) represents the sensitivity of \(D_k (S_k)\) to parameter \(p_m\). Since \(\tau\) in (3.5) and (3.6) is a function of \(\vec{R}\) and \(\vec{C}\), \(\frac{\partial \{D_k, S_k\}}{\partial p_m}\) can be expanded by the chain rule \([98]\) as:

\[ \frac{\partial \{D_k, S_k\}}{\partial p_m} = \sum_{r \in \vec{R}} \frac{\partial \{D_k, S_k\}}{\partial r} \frac{\partial r}{\partial p_m} + \sum_{c \in \vec{C}} \frac{\partial \{D_k, S_k\}}{\partial c} \frac{\partial c}{\partial p_m} \]  

(3.8)

The trend of variations \(\partial r/\partial p_m\) and \(\partial c/\partial p_m\) of the basic logic gates is obtained from TCAD simulations, and stored in the FinFET design library.

Suppose that there are \(l\) gates on the critical path. The mean and variance of the critical path delay \((d_{crit})\) are given by:

\[ \mu_{crit} = \sum_{k=1}^{l} d_{k,nom} \]  

(3.9)

\[ Var(d_{crit}) = \sum_{k=1}^{l} Var(d_k) + 2 \sum_{i \neq j} Cov(d_i, d_j) \]  

(3.10)

To calculate \(Cov(d_i, d_j)\), the covariance of the PV parameters and \(\Delta s_k\)'s are required.

The rectangular grid-based method facilitates the calculation of the covariance of PV parameters. However, the calculation of the covariance of \(\Delta s_k\)'s is not straightforward and requires further parameter transformation.

The idea behind parameter transformation is to transform \(\Delta s_k\)'s to expressions in PV parameters \(\Delta p_m\)'s. We next consider only a single parameter \(\Delta p_m\) for illustration.
Assuming a step input, the delay of gates 1 and 2 (where gate 1 feeds gate 2) can be expressed as:

\[ d_1 = d_{1,nom} + \frac{\partial D_1}{\partial p_m} \Delta p_m \]  

(3.11)

\[ s_1 = s_{1,nom} + \frac{\partial S_1}{\partial p_m} \Delta p_m = s_{1,nom} + \Delta s_1 \]  

(3.12)

Based on (3.7), (3.11) and (3.12), \( d_1 \) is derived as:

\[ d_2 = d_{2,nom} + \frac{\partial D_2}{\partial p_m} \Delta p_m + \frac{\partial D_2}{\partial s_1} \frac{\partial S_1}{\partial p_m} \Delta p_m \]  

(3.13)

Assuming the time constant \( \tau \) of gate \( k \) is equal to \( RC \) and \( \Delta s_1 \) is expressed as \( \frac{\partial S_1}{\partial p_m} \Delta p_m \), we further generalize \( d_k \) as:

\[ d_k = d_{k,nom} + \frac{\partial D_k, total}{\partial p_m} \Delta p_m \]  

(3.14)

where \( \frac{\partial D_k, total}{\partial p_m} \Delta p_m \) describes the total delay deviation from \( d_{k,nom} \). Based on (3.5), (3.6), (3.7), and (3.8), it can be expressed as follows [88]:

\[ \frac{\partial D_k, total}{\partial p_m} = \frac{1}{2 \cdot d_{k,nom}} \cdot \left\{ \frac{\partial R}{\partial p_m} \cdot (2 \cdot \alpha_d \cdot R \cdot C^2 + A \cdot s_{k-1} \cdot C) \ight. 
\] 

\[ + \frac{\partial C}{\partial p_m} \cdot (2 \cdot \alpha_d \cdot R^2 \cdot C + A \cdot s_{k-1} \cdot R) \]

\[ + \frac{\partial s_{k-1}}{\partial p_m} \cdot A \cdot R \cdot C \} \]  

(3.15)

where \( \frac{\partial s_k}{\partial p_m} \) can be expressed as [88]:

\[ \frac{\partial s_k}{\partial p_m} = \frac{S_{base}}{\tau} \cdot (\alpha_s + X^2) \cdot \left( R \cdot \frac{\partial C}{\partial p_m} + C \cdot \frac{\partial R}{\partial p_m} \right) 
\] 

\[ + 2 \cdot S_{base} \cdot X \cdot \gamma_s \cdot \frac{s_{k-1}/\partial p_m}{s_{k-1} + \beta_s \cdot S_{base}} \] 

\[ - 2 \cdot S_{base} \cdot X \cdot \gamma_s \cdot \frac{s_{k-1}/\partial p_m}{s_{k-1} + \beta_s \cdot S_{base}} \cdot \left[ \frac{s_{k-1} \cdot (R \cdot \frac{\partial C}{\partial p_m} + C \cdot \frac{\partial R}{\partial p_m})}{\tau} \right] \]

(3.16)
The delay of gate $k$ is now represented by PV parameters only. Based on (3.14), (3.7) can be rewritten as:

$$d_k = d_{k,nom} + \sum_{p_m \in \vec{P}_m} F(k, p_m) \Delta p_m$$

(3.17)

with mean $d_{k,nom}$ and variance expressed as follows:

$$Var(d_k) = \sum_{p_m \in \vec{P}_m} (F(k, p_m))^2 Var(\Delta p_m)$$

(3.18)

The covariance of $d_i$ in grid $m$ and $d_j$ in grid $n$ can be represented as:

$$Cov(d_i, d_j) = \sum_{p_m \in \vec{P}_m \atop p_n \in \vec{P}_n} (F(i, p_m) F(j, p_n)) Cov(\Delta p_m, \Delta p_n)$$

(3.19)

where $Cov(\Delta p_m, \Delta p_n)$ is calculated in a similar manner as (3.3). By applying (3.18) and (3.19) to (3.10), we are able to estimate the variance of the critical path delay in just one round of simulation. Hence, the above method enables quick estimation of the delay spread.

### 3.2.2 Leakage variation model

The leakage current of a logic gate or memory cell is modeled as a lognormal RV. FinCANNON uses a leakage variation model generalized from the macromodeling method in [77]. The macromodeling method is used instead of simple leakage estimation because of the following reasons:

- Cache has a regular structure, which consists of memory cells and logic gates. As a result, deriving the leakage power of the cache from these components is reasonable.

- Leakage power consumption is estimated accurately.
• If the cache structure is changed, simple estimation of the leakage can lead to errors. Leakage power estimated from fundamental components maintains accuracy.

• Leakage power is calculated at the subarray level only. The leakage power of larger components (e.g., cache banks) is calculated by the architectural macromodel methodology introduced in Section 4.1.1.

The macromodels of logic gates and memory cells are formulated by empirical parameters extracted from TCAD device simulations. These parameters are obtained by fitting the simulated data points with the lowest-degree polynomial or exponential expressions that yield the least residual. Each type of logic gate or memory cell is calibrated with a set of empirical parameters. We generalize the leakage macromodel as:

\[
I_{\text{Leak}}(k, \mathbf{p}_m) = I_{\text{Leak}}(k, \text{Base}) \cdot e^{Q(p_m)}
\]

\[
= I_{\text{Leak}}(k, \text{Base}) \cdot e^{Q(p_m)}
\]

(3.20)

where \(I_{\text{Leak}}(k, \mathbf{p}_m)\) is the leakage current of gate \(k\), and \(I_{\text{Leak}}(k, \text{Base})\) and \(a_1 \sim a_7\) are empirical constants (\(a_1 \sim a_7\) are associated with PV parameter \(p_m\)). Polynomial \(Q(p_m)\) determines the variation trend. Eq. (3.20) models the leakage variation trend in an exponential fashion for all the PV parameters in \(\mathbf{p}_m\). If higher-order polynomials are used in \(Q(p_m)\), the enhanced leakage macromodel can be applied to a wider range of parameter variations. Based on (3.20), the leakage power \(p_k\) of a logic gate \(k\) is given by:

\[
p_k = P_k(I_{\text{Leak}}(k, \mathbf{p}_m), V_{DD}) = I_{\text{Leak}}(k, \mathbf{p}_m) \cdot V_{DD}
\]

(3.21)
where $P_k$ is the leakage power function of gate $k$, $m$ the grid where gate $k$ resides, and $V_{DD}$ the supply voltage for gate $k$.

Since $P_k$ is modeled as a lognormal RV, a statistical approach for obtaining the sum of $P_k$’s requires a different mechanism. Although the sum of lognormal RVs is theoretically not known to have a closed form, Wilkinson’s method gives a fairly accurate approximation to this problem \[158\]. Due to the limitation of space, we do not elaborate upon the detailed mathematics here, but refer interested readers to \[149\]. A direct application of Wilkinson’s method to leakage power analysis requires a complexity of $O(N^2)$, where $N$ is the number of gates in the circuit. Hence, the computation time increases drastically for large circuits.

To overcome the above problem, instead of computing leakage power on a gate-by-gate basis, we first divide the circuit by grids using the grid plan discussed previously. Next, inside each grid, the leakage power of all gates are summed up to give the leakage power of the grid, assuming a correlation of 1. Finally, leakage power of the circuit is obtained by applying Wilkinson’s method to all the grids in the same fashion. In this way, the complexity is reduced from $O(N^2)$ to $O(n^2)$, where $n$ is the total number of grids, which is typically much smaller than $N$. This has minimal impact on accuracy.

### 3.2.3 Temperature macromodel

The FinFET design library models temperature variation using a macromodel approach. Given a circuit-level parameter $G(P_m)$, its temperature macromodel is generalized from \[88\] as follows:

$$G(P_m, T_{op}) = G(P_m) \cdot G_{base} \cdot e^{G_{temp}(T_{op})}$$

where $G(P_m, T_{op})$ denotes the macromodel that takes the effect of temperature into account, $G_{base}$ is an empirical constant, $T_{op}$ is the operating temperature, and $G_{temp}(T_{op})$...
is the polynomial that determines the variation trend. They are obtained by regression in which simulated data points are modeled with the lowest-degree exponential expressions that yield the least residual. The FinFET design library contains one such temperature macromodel for each type of logic gate and memory cell. If higher-order polynomials are used in $G_{temp}(T_{op})$, the macromodel can be applied to a wider range of temperature variation.

### 3.3 Voltage Generators

Fig. 3.4(a) shows the block diagram of the voltage generator. The voltage generator consists of the following major blocks:

**Resistor tree:** This block consists of a series of resistors. These resistors divide the voltage range ($V_H$ to $V_L$) and provide the required reference voltage levels. The resistance values are designed to be high so that the power consumed by the current flowing through it is minimized.

**Voltage selection net:** The function of the voltage selection net is to select a voltage level from the resistor tree and feed it to the unity-gain amplifier. Parallel pass transistors are used for this purpose, each connected to a tap in the resistor tree, providing various voltage levels to the unity-gain amplifier. Only one pass transistor is turned on at a time.

**Unity-gain amplifier:** The unity gain amplifier drives the back gates of FinFETs to the reference voltage level received from the voltage selection net. The transistor sizes used in the amplifier are scaled appropriately to enable this. The size of the driving stage of the amplifier is determined by the capacitance (i.e., number of back gates of the FinFETs) that the amplifier needs to drive. If the amplifier is to drive a large component, it is necessary to either increase the size of the output stage, or use multiple amplifiers in the component. Two unity-gain amplifiers are required: one
Digital controller: This unit provides the digital selection signals for the voltage selection net. These signals are determined by the flow control mechanism.

Inspired by the op-amp designs in [40, 159], we have laid out voltage generators for FinFETs at the 22nm technology node. Fig. 3.4(b) shows the layout of a basic voltage generator optimized for the nFinFETs in the mat with four 64×64 subarrays. A mat is the basic building block of a cache bank in CACTI-PVT. The size of the voltage generator is 1.225µm × 2.745µm, including the resistor tree, voltage selection net, and the op-amp voltage buffer for nFinFETs. We optimize the design for fast transition (within 1ns or one clock cycle at 1GHz) from normally-biased ($V_{HI} = 0.9V$)
Table 3.1: Variable PVT parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Nominal value</th>
<th>Range</th>
<th>Step Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate length, $L_G$ (nm)</td>
<td>20</td>
<td>[18,22]</td>
<td>0.4</td>
</tr>
<tr>
<td>Silicon thickness, $T_{SI}$ (nm)</td>
<td>10</td>
<td>[9,11]</td>
<td>0.2</td>
</tr>
<tr>
<td>Oxide thickness, $T_{OX}$ (nm)</td>
<td>1</td>
<td>[0.9,1.1]</td>
<td>0.02</td>
</tr>
<tr>
<td>nFinFET workfunction, $\Phi_N$ (eV)</td>
<td>4.4</td>
<td>[4.38,4.42]</td>
<td>0.004</td>
</tr>
<tr>
<td>pFinFET workfunction, $\Phi_P$ (eV)</td>
<td>4.8</td>
<td>[4.78,4.82]</td>
<td>0.004</td>
</tr>
<tr>
<td>Supply voltage, $V_{DD}$ (V)</td>
<td>0.9</td>
<td>[0.8,1.0]</td>
<td>0.02</td>
</tr>
<tr>
<td>Temperature, $T_{op}$ (K)</td>
<td>N.A.</td>
<td>[298,398]</td>
<td>10</td>
</tr>
</tbody>
</table>

and $V_{LOW} = 0V$) voltage levels to reverse-biased voltage levels ($V_{HI} = 1.1V$ and $V_{LOW} = -0.2V$), and vice versa. To accomplish this, the amplifier is scaled to tackle the load capacitance, which is equal to the sum of the interconnect capacitances and the back-gate capacitances of the FinFETs in the targeted circuit component. A range of voltage levels ($-0.2V \sim 1.1V$) can be generated and selected by digitally-controlled inputs.

### 3.4 Experimental Results

In this section, we show experimental results for the FinFET design library. We evaluate the impact of PVT variations on delay and power of logic gates and circuits. We validate our delay, leakage, and temperature models with QMC simulations. Finally, we explore the leakage power of four different types of FinFET memory cells. Tables 3.1, 3.6, 3.7, 3.9, and 3.10 are also available in [88]. Tables 3.2, 3.3, 3.4, and 3.5 and Figs. 3.8 and 3.10 are expanded and enhanced from those in [88].

#### 3.4.1 Simulation setup

In our simulations, we assume a normal distribution of PV parameters and set the $3\sigma$ value of the distribution to 10% of the nominal value. Table 3.1 lists the parameters
that are subject to variations, their nominal values, ranges of variation, and step size.

The ranges of variation correspond to \([-3\sigma, 3\sigma]\) values of the PV parameters. \(T_{op}\) is not normally distributed, and is separately specified. If not specifically stated, \(T_{op}\) is assumed to be 298K. The hardware platform used in the simulation is a 112-node Intel Westmere computer cluster. Each processor has 4GB of RAM and operates at 2.67 GHz. We use SG-mode logic gates to validate the delay and leakage models, and compare results with the IG- and ASG-mode cases. Fig. 3.5 shows the layouts of SG-mode logic gates. We explore four different designs of memory cells, including 4T [60], pass-gate feedback (PGFB) 6T [62], row-based back-gate biasing (RBGB) 6T [42], and 8T SRAM cells [61]. Figs. 3.6(a) and 3.6(b) show the circuits and custom layouts of those FinFET SRAMs cells, respectively.

### 3.4.2 Impact of PVT variations on FinFET logic gates

We use QMC simulations to characterize the effects of PV variations on delay and leakage of logic gates. Among the process parameters, the impact of \(L_G\), \(T_{SI}\), \(T_{OX}\), \(\Delta\Phi_{GN}\), and \(\Delta\Phi_{GP}\) on logic gate performance has been experimentally shown to be the most prominent [77, 78, 96], where \(\Delta\Phi_{GN}\) (\(\Delta\Phi_{GP}\)) is the workfunction difference of both the front and back gate electrodes of nFinFETs (pFinFETs) with respect to intrinsic silicon. Varying these PV parameters directly affects the \(V_{th}\) of FinFETs,
which, in turn, results in variability in the delay and $I_{\text{Leak}}$ of logic gates. Fig. 3.7 illustrates the delay and leakage power spread of circuit s27 from the ISCAS’89 [160, 161] benchmark suite, with $L_G$, $T_{SI}$, $T_{OX}$, $\Delta \Phi_{GN}$, $\Delta \Phi_{GP}$, and $V_{DD}$ individually varying normally, and $3\sigma/\mu = 10\%$. The temperature is 298K. The QMC tool [102] used in the simulations is based on Sobol’s low-discrepancy sequence, which avoids the sample clustering problem encountered in Monte Carlo (MC) simulation. From Fig. 3.7(a), we identify $L_G$, $T_{SI}$, and $V_{DD}$ as the main parameters that affect gate delay. From Fig. 3.7(b), we observe that $L_G$, $T_{SI}$, and $\Delta \Phi_{GN}$ affect leakage power spreads significantly.

Figure 3.6: FinFET SRAM cells
Figure 3.7: QMC parameter characterization of s27
Impact of PV variations on circuit-level parameters

Tables 3.2, 3.3, 3.4, and 3.5 show the effect of PV variations on circuit-level parameters, including $C_{in}$, $C_{out}$, $R_{out}$, and $I_{Leak}$. The gate size is X8. In general, increasing $L_G$ leads to larger gate area, and hence, larger $C_{in}$. Increasing $L_G$ also increases $R_{out}$ and reduces $I_{Leak}$. This is because the effect of SCEs (which are mainly attributable to DIBL) is alleviated. Additionally, the increased gate length provides larger resistance and, hence, reduces the leakage current. A 1.6nm increase in $L_G$ from its nominal value of 20nm can change $I_{Leak}$ by 33.3% for the NOR gate. Variation in $T_{SI}$ does not have a significant impact on $C_{in}$, $C_{out}$, and $R_{out}$. However, a 3$\sigma$ increase in $T_{SI}$ from its nominal values may increase $I_{Leak}$ by up to 33.7% (NOR). The increased $I_{Leak}$ is caused by the increased cross-section of the channel. Variation in $T_{OX}$ affects $C_{in}$ significantly. An 8% increase in $T_{OX}$ from its nominal value may change $C_{in}$ by up to 6.7% (INV). Variations in $\Delta \Phi_{GN}$ and $\Delta \Phi_{GP}$ have little effect on $C_{in}$, $C_{out}$, and $R_{out}$. However, less than 8% variation in $\Delta \Phi_{GN}$ can change $I_{Leak}$ by up to 63.9% in the NAND gate case. Although variation in $\Delta \Phi_{GP}$ has a smaller impact on $I_{Leak}$ than $\Delta \Phi_{GN}$, $I_{Leak}$ deviates by more than 17.3% for NOR with a variation of 8% in $\Delta \Phi_{GP}$ from its nominal value. The impact of variation in $\Delta \Phi_{GN}$ is generally greater than that in $\Delta \Phi_{GP}$ because electrons have higher mobility than holes.

Impact of temperature variation on circuit-level parameters

Fig. 3.8 shows the effect of temperature variation on circuit-level parameters. It can be seen that as $T_{op}$ increases, $R_{out}$ and $I_{Leak}$ increase as well. At 398K, $R_{out}$ and $I_{Leak}$ of INV X8 are 1.17× and 27.37× larger as compared to those at 298K, respectively. On the other hand, both $C_{in}$ and $C_{out}$ do not show significant change as $T_{op}$ increases.
### Table 3.2: Impact of PV variations on $C_{in}$

<table>
<thead>
<tr>
<th>PV parameter</th>
<th>Gate</th>
<th>$C_{in}$ value (fF)</th>
<th>-8%</th>
<th>-4%</th>
<th>Nominal</th>
<th>4%</th>
<th>8%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_G$</td>
<td>INV</td>
<td>1.68 1.75 1.77 1.78 1.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.60 0.60 0.61 0.62 0.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>2.88 2.95 2.97 2.97 3.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{SI}$</td>
<td>INV</td>
<td>1.80 1.78 1.77 1.78 1.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.61 0.61 0.61 0.62 0.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>2.97 2.99 2.97 2.99 3.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{OX}$</td>
<td>INV</td>
<td>1.91 1.85 1.77 1.72 1.67</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.64 0.62 0.61 0.60 0.59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>3.14 3.06 2.97 2.92 2.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta \Phi_{GN}$</td>
<td>INV</td>
<td>1.79 1.80 1.77 1.77 1.76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.61 0.61 0.61 0.61 0.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>3.01 2.98 2.97 2.96 2.97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta \Phi_{GP}$</td>
<td>INV</td>
<td>1.74 1.76 1.77 1.81 1.82</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.61 0.61 0.61 0.61 0.61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>2.98 2.96 2.97 2.98 3.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{DD}$</td>
<td>INV</td>
<td>1.66 1.73 1.77 1.82 1.83</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.60 0.60 0.61 0.62 0.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>2.89 2.92 2.97 2.99 3.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3.3: Impact of PV variations on $C_{out}$

<table>
<thead>
<tr>
<th>PV parameter</th>
<th>Gate</th>
<th>$C_{out}$ value (fF)</th>
<th>-8%</th>
<th>-4%</th>
<th>Nominal</th>
<th>4%</th>
<th>8%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_G$</td>
<td>INV</td>
<td>0.43 0.43 0.42 0.41 0.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.48 0.49 0.49 0.49 0.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.65 0.65 0.63 0.60 0.59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{SI}$</td>
<td>INV</td>
<td>0.42 0.42 0.42 0.43 0.43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.48 0.49 0.49 0.49 0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.62 0.62 0.63 0.64 0.65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{OX}$</td>
<td>INV</td>
<td>0.44 0.43 0.42 0.42 0.41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.51 0.50 0.49 0.48 0.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.65 0.64 0.63 0.63 0.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta \Phi_{GN}$</td>
<td>INV</td>
<td>0.43 0.43 0.42 0.42 0.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.49 0.49 0.49 0.49 0.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.65 0.63 0.63 0.63 0.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta \Phi_{GP}$</td>
<td>INV</td>
<td>0.42 0.42 0.42 0.43 0.43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.50 0.49 0.49 0.49 0.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.64 0.63 0.63 0.63 0.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{DD}$</td>
<td>INV</td>
<td>0.41 0.42 0.42 0.44 0.44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.48 0.49 0.49 0.49 0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.64 0.63 0.63 0.64 0.64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 3.4: Impact of PV variations on $R_{out}$

<table>
<thead>
<tr>
<th>PV parameter</th>
<th>Gate</th>
<th>$R_{out}$ value (kΩ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_G$</td>
<td>INV</td>
<td>0.54, 0.56, 0.60, 0.64, 0.66</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>0.93, 0.99, 1.06, 1.12, 1.17</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.54, 0.57, 0.62, 0.66, 0.68</td>
</tr>
<tr>
<td>$T_{SI}$</td>
<td>INV</td>
<td>0.62, 0.61, 0.60, 0.59, 0.58</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>1.09, 1.08, 1.06, 1.04, 1.03</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.63, 0.62, 0.62, 0.61, 0.60</td>
</tr>
<tr>
<td>$T_{OX}$</td>
<td>INV</td>
<td>0.58, 0.59, 0.60, 0.61, 0.61</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>1.04, 1.05, 1.06, 1.07, 1.08</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.60, 0.61, 0.62, 0.62, 0.63</td>
</tr>
<tr>
<td>$\Delta \Phi_{GN}$</td>
<td>INV</td>
<td>0.59, 0.59, 0.60, 0.61, 0.61</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>1.05, 1.05, 1.06, 1.07, 1.08</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.60, 0.61, 0.62, 0.62, 0.62</td>
</tr>
<tr>
<td>$\Delta \Phi_{GP}$</td>
<td>INV</td>
<td>0.60, 0.60, 0.60, 0.60, 0.60</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>1.05, 1.06, 1.06, 1.06, 1.06</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.61, 0.62, 0.62, 0.62, 0.62</td>
</tr>
<tr>
<td>$V_{DD}$</td>
<td>INV</td>
<td>0.63, 0.63, 0.60, 0.59, 0.59</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>1.10, 1.08, 1.06, 1.04, 1.03</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>0.64, 0.63, 0.62, 0.61, 0.60</td>
</tr>
</tbody>
</table>

### Table 3.5: Impact of PV variations on $I_{Leak}$

<table>
<thead>
<tr>
<th>PV parameter</th>
<th>Gate</th>
<th>$I_{Leak}$ value (nA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_G$</td>
<td>INV</td>
<td>35.98, 31.60, 27.14, 21.93, 17.98</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>38.45, 33.94, 29.34, 23.97, 19.87</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>41.55, 36.57, 31.49, 25.55, 21.03</td>
</tr>
<tr>
<td>$T_{SI}$</td>
<td>INV</td>
<td>20.44, 23.75, 27.14, 31.73, 36.26</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>22.41, 25.89, 29.34, 34.14, 38.47</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>23.66, 27.47, 31.49, 36.79, 42.09</td>
</tr>
<tr>
<td>$T_{OX}$</td>
<td>INV</td>
<td>24.32, 25.69, 27.14, 28.67, 30.29</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>26.45, 27.86, 29.34, 30.90, 32.55</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>28.20, 29.80, 31.49, 33.28, 35.16</td>
</tr>
<tr>
<td>$\Delta \Phi_{GN}$</td>
<td>INV</td>
<td>43.21, 34.15, 27.14, 21.72, 17.53</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>48.10, 37.52, 29.34, 23.02, 18.13</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>47.56, 38.50, 31.49, 26.07, 21.89</td>
</tr>
<tr>
<td>$\Delta \Phi_{GP}$</td>
<td>INV</td>
<td>25.74, 26.35, 27.14, 28.17, 29.51</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>28.65, 28.95, 29.34, 29.85, 30.51</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>28.29, 29.68, 31.49, 33.85, 36.93</td>
</tr>
<tr>
<td>$V_{DD}$</td>
<td>INV</td>
<td>26.10, 26.61, 27.14, 27.67, 28.20</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>28.33, 28.83, 29.34, 29.85, 30.36</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>30.26, 30.87, 31.49, 32.11, 32.73</td>
</tr>
</tbody>
</table>
3.4.3 Validation of delay and leakage models

We validate our delay and leakage models under temperature variation. The validation is conducted in two phases. First, we show that the residuals of our models are small. The residual is the difference between the sample point and the estimated value generated by the model. Next, we perform parameter value sampling for QMC simulations with Gaussian-distributed $L_G$, $T_{SI}$, $T_{OX}$, $\Delta \Phi_{GN}$, $\Delta \Phi_{GP}$, and $V_{DD}$, and uniformly distributed $T_{op}$.

We simulate 100 samples for each of the three types of logic gates and four gate sizes and report the statistical errors. The statistical error corresponds to the difference between the QMC simulation result and the estimated value generated by the model.

Delay model validation

To validate the delay model, we first validate the output slope ($S_{out}$) and delay ($d_{gate}$)
Table 3.6: Residual and statistical errors of the delay model

<table>
<thead>
<tr>
<th>Gate</th>
<th>Size</th>
<th>Residual (%)</th>
<th>Statistical error: QMC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$S_{out}$</td>
<td>$d_{gate}$</td>
</tr>
<tr>
<td>INV</td>
<td>1</td>
<td>1.2</td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.8</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3.9</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3.7</td>
<td>4.8</td>
</tr>
<tr>
<td>NAND</td>
<td>1</td>
<td>2.8</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.9</td>
<td>3.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.7</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2.6</td>
<td>3.6</td>
</tr>
<tr>
<td>NOR</td>
<td>1</td>
<td>1.6</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.9</td>
<td>4.3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4.6</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>6.0</td>
<td>7.8</td>
</tr>
</tbody>
</table>

models of each gate, as shown in Table 3.6. In addition to obtaining the QMC samples mentioned earlier, we add samples for the input slope ($S_{in}$) and load capacitance ($C_{load}$) based on uniform distribution and evaluate the statistical error of the proposed timing model. On an average, the residual errors for $S_{out}$ and $d_{gate}$ are 3.0% and 4.0%, respectively. The QMC test shows that the proposed delay model as compared to the QMC results has an average statistical error of only 3.4% and 4.4% for $S_{out}$ and $d_{gate}$, respectively.

**Leakage model validation under temperature variation**

Table 3.7 reports the simulation results for leakage model validation. The third column reports the residual and the fourth column the statistical error. On an average, the residual and statistical errors are 3.0% and 3.1%, respectively. This seems to offer a good efficiency-accuracy tradeoff. If a higher level of accuracy is desired, the macromodel can be extended to higher-order polynomials.

### 3.4.4 Comparison of FinFET logic gate styles

Table 3.8 compares $C_{in}$, $C_{out}$, $R_{out}$, and $P_{Leak}$ of a minimum-sized inverter for three types of logic styles. The IG-mode inverter is biased at $V_{HI} = 1.1V$ and $V_{LOW} = 59$
Table 3.7: Residual and statistical errors of the leakage model

<table>
<thead>
<tr>
<th>Gate</th>
<th>Size</th>
<th>$I_{leak,298K\sim398K}$ residual (%)</th>
<th>$I_{leak,QMC-100}$ statistical error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>1</td>
<td>2.8</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.9</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.8</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4.0</td>
<td>3.6</td>
</tr>
<tr>
<td>NAND</td>
<td>1</td>
<td>2.7</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.7</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.7</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3.7</td>
<td>3.3</td>
</tr>
<tr>
<td>NOR</td>
<td>1</td>
<td>2.7</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.6</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.5</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3.9</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Table 3.8: Comparison of FinFET logic gate styles

<table>
<thead>
<tr>
<th>Gate style</th>
<th>$C_{in}$ (fF)</th>
<th>$C_{out}$ (fF)</th>
<th>$R_{out}$ (kΩ)</th>
<th>$P_{Leak}$ (nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG</td>
<td>0.16</td>
<td>0.06</td>
<td>4.51</td>
<td>1.44</td>
</tr>
<tr>
<td>IG</td>
<td>0.06</td>
<td>0.04</td>
<td>15.16</td>
<td>0.10</td>
</tr>
<tr>
<td>ASG</td>
<td>0.11</td>
<td>0.05</td>
<td>7.04</td>
<td>0.01</td>
</tr>
</tbody>
</table>

$-0.2V$. The SG-mode inverter has the highest $C_{in}$ due to the shorted double gates. Its $R_{out}$ is the lowest among the three due to its high on-current. Its $P_{Leak}$ is also the highest. The $C_{in}$ of the IG-mode inverter is the smallest among the three, because only one of the two gates is used for logic operation. Its $R_{out}$ is the highest due to its lower on-current. The ASG-mode inverter provides ultra-low off-current, which is even lower than that of the IG-mode inverter. It also provides relatively low $R_{out}$, thus its on-current is only slightly lower than that of the SG-mode inverter. These results imply two facts in FinFET-based circuit design. First, SG-mode logic gates are necessary when circuit delay is the most important concern. Second, ASG-mode logic gates can provide better performance in terms of both delay and power than IG-mode logic gates. IG-mode logic gates also occupy larger area and, hence, should be substituted with ASG-mode logic gates whenever possible.
3.4.5 Leakage power of FinFET memory cells

Fig. 3.9 compares leakage power of FinFET memory cells for various temperatures. In Fig. 3.9(a), we compare four different types of memory cells. The IG-mode FinFETs in the RBGB cell are reverse-biased by $-0.2V$. In Fig. 3.9(b), we vary the reverse-bias voltage of the RBGB cell from $-0.1V$ to $-0.3V$ and observe the change in leakage power. In all cases, the leakage power increases exponentially as the temperature increases. It can be seen that the RBGB cell has the lowest leakage power as compared to the other three types of cells. This is due to the increased $V_{th}$ provided by reverse biasing. The 4T SRAM cell has the second lowest leakage power. At low temperatures, the 8T SRAM cell has the highest leakage power. This is caused by the extra read path in the 8T cell. At high temperatures, the leakage power of the PGFB cell exceeds that of the 8T cell. This is because the p-FinFETs in the PGFB cell are connected in SG mode, while they are connected in IG mode in the 8T SRAM cell. The leakage power of SG-mode FinFETs increases faster than that of the IG-mode FinFETs as the temperature increases. The read path in the 8T cell has two serially-connected FinFETs and, hence, does not outweigh the effects of the SG-mode FinFETs in the PGFB cell. As a result, the leakage power of the PGFB cell is the highest at 398K. In Fig. 3.9(b) it can be seen that the leakage power decreases as the bias voltage of the cell increases. When biased at $-0.1V$, the leakage power at 398K is $2.85 \times$ as compared to the $-0.3V$ case. This is because $V_{th}$ increases as the reverse bias increases.

3.4.6 Impact of PVT variations on FinFET circuits

Next, we show the impact of PVT variations on FinFET circuits. We first illustrate the impact of PV variations on delay and power. Then we show the trends in delay and power when increasing $T_{op}$. Finally, we report simulation results for the ISCAS’89 [160, 161] benchmark suite based on our models, and compare them with
Figure 3.9: Memory cell leakage vs. temperature

(a) Comparison of different memory cell styles

(b) Varying memory cell bias voltage
Table 3.9: Impact of temperature variation on $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ for s38584

<table>
<thead>
<tr>
<th>Temperature (K)</th>
<th>Delay (ps)</th>
<th>$P_{Leak}$ ($\mu W$)</th>
<th>$P_{Dyn}$ ($\mu W$)</th>
<th>$\frac{\Delta P_{Dyn}}{P_{Dyn}}$ %</th>
</tr>
</thead>
<tbody>
<tr>
<td>298</td>
<td>1005.6</td>
<td>125.1</td>
<td>1244.1</td>
<td>10.1</td>
</tr>
<tr>
<td>308</td>
<td>1010.8</td>
<td>189.2</td>
<td>1246.3</td>
<td>15.2</td>
</tr>
<tr>
<td>318</td>
<td>1016.0</td>
<td>280.8</td>
<td>1248.6</td>
<td>22.5</td>
</tr>
<tr>
<td>328</td>
<td>1021.2</td>
<td>409.1</td>
<td>1250.9</td>
<td>32.7</td>
</tr>
<tr>
<td>338</td>
<td>1026.4</td>
<td>585.0</td>
<td>1253.1</td>
<td>46.7</td>
</tr>
<tr>
<td>348</td>
<td>1031.6</td>
<td>821.1</td>
<td>1255.4</td>
<td>65.4</td>
</tr>
<tr>
<td>358</td>
<td>1036.8</td>
<td>1131.3</td>
<td>1257.7</td>
<td>89.9</td>
</tr>
<tr>
<td>368</td>
<td>1042.0</td>
<td>1529.7</td>
<td>1259.9</td>
<td>121.4</td>
</tr>
<tr>
<td>378</td>
<td>1047.2</td>
<td>2030.4</td>
<td>1262.2</td>
<td>160.9</td>
</tr>
<tr>
<td>388</td>
<td>1052.4</td>
<td>2645.1</td>
<td>1264.5</td>
<td>209.2</td>
</tr>
<tr>
<td>398</td>
<td>1057.7</td>
<td>3382.6</td>
<td>1266.8</td>
<td>267.0</td>
</tr>
</tbody>
</table>

those obtained from MC simulations. These benchmarks represent a wide variety of problem domains and are the most frequently cited of all benchmarks [161]. We use an operating frequency of 500MHz.

Impact of PV variations on delay and power

Fig. 3.10 illustrates the impact of PV variations on circuit delay ($D_{Ckt}$), leakage power ($P_{Leak}$), and dynamic power ($P_{Dyn}$) of benchmark s38584 (the largest one from the ISCAS’89 benchmark suite). It can be seen that variations in $L_G$ have the most impact on $D_{Ckt}$, resulting in a 7.6% variation over the $[-3\sigma, 3\sigma]$ range. On the other hand, $\Delta \Phi_{GN}$, $\Delta \Phi_{GP}$, and $V_{DD}$ have less than 1% impact on $D_{Ckt}$. In the case of $P_{Leak}$, $\Delta \Phi_{GN}$ has the most impact, causing a variation of as much as 71.0%. Last, $P_{Dyn}$ is mostly impacted by $V_{DD}$, since $P_{Dyn}$ is linearly dependent on $V_{DD}^2$. $T_{OX}$ also has a significant impact on $P_{Dyn}$ as compared to the other PV parameters, resulting in a 7.3% variation over the range.

Impact of temperature variation on delay and power

Table 3.9 shows the impact of temperature variation on $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ of benchmark s38584. Temperature significantly affects both $D_{Ckt}$ and $P_{Leak}$, however,
Figure 3.10: Impact of PV variations on $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ for s38584 ($T_{op} = 298K$)
has little impact on $P_{Dyn}$. It can been seen that as $T_{op}$ increases, both $D_{Ckt}$ and $P_{Leak}$ increase. $D_{Ckt}$ increases by 5.2% as $T_{op}$ increases from 298K to 398K. The leakage power increases exponentially. The value of $P_{Leak}$ at 398K is $27.04 \times$ larger as compared to that at 298K. Meanwhile, the ratio of $P_{Leak}$ to $P_{Dyn}$ increases from 10.1% to 267.0%. As a result, $P_{Dyn}$ dominates power consumption at low temperatures, while $P_{Leak}$ contributes most of the power consumption at high temperatures.

**Validation of the PVT variation models**

We validate the PVT variation models by comparing them with MC simulation results. Table 3.10 compares $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ between the MC simulation results and the proposed PVT variation models for various ISCAS’89 benchmarks [160, 161]. We perform 10,000 MC simulations for each benchmark circuit. The CPU time ratio ($CPU_{ratio}$) of a circuit is the ratio of the CPU time required to perform the MC simulation to that required by the PVT variation models. For each MC simulation result, we show the mean and STD of $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$, respectively. We then show the mean values obtained from the PVT variation models, and evaluate the errors of the evaluated STDs with respect to those obtained from the MC simulations. The STD errors of $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ are shown in Fig. 3.11. It can be seen that the proposed evaluation methodology reduces CPU time significantly. The mean values in both cases match closely. The errors in STDs are typically small except for the $D_{Ckt}$ error for s38417. This is due to the Gaussian approximation used for the statistical max operation.

### 3.5 Chapter Summary

In this chapter, we presented the FinFET design library, which consists of a FinFET logic library and a FinFET memory library. The former contains macromodels of various FinFET logic gates, while the latter contains macromodels of FinFET memory
Table 3.10: Validation of the PVT variation models

<table>
<thead>
<tr>
<th>Benchmark</th>
<th># Gates</th>
<th>( CPUG)</th>
<th>( CPUG) ratio</th>
<th>Monte-Carlo (MC)</th>
<th>PVT variation models</th>
<th>( STD_{\text{MC}} )</th>
<th>( STD_{\text{MC}} )</th>
<th>( % )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>( D_{\text{Ckt}} ) (ps)</td>
<td>( P_{\text{Leak}} ) (( \mu W ))</td>
<td>( P_{\text{Dyn}} ) (( \mu W ))</td>
<td>( D_{\text{Ckt}} ) (ps)</td>
<td>( P_{\text{Leak}} ) (( \mu W ))</td>
<td>( P_{\text{Dyn}} ) (( \mu W ))</td>
</tr>
<tr>
<td>s38584</td>
<td>32270</td>
<td>55.6</td>
<td>1017.3</td>
<td>19.9</td>
<td>281.4</td>
<td>66.6</td>
<td>1252.0</td>
<td>82.4</td>
</tr>
<tr>
<td>s38417</td>
<td>30332</td>
<td>52.5</td>
<td>304.9</td>
<td>8.1</td>
<td>200.2</td>
<td>65.5</td>
<td>1163.9</td>
<td>79.2</td>
</tr>
<tr>
<td>s35952</td>
<td>625.2</td>
<td>625.2</td>
<td>1351.4</td>
<td>33.1</td>
<td>294.9</td>
<td>69.9</td>
<td>1035.2</td>
<td>70.9</td>
</tr>
<tr>
<td>s15850</td>
<td>13408</td>
<td>59.7</td>
<td>333.9</td>
<td>12.2</td>
<td>129.5</td>
<td>30.6</td>
<td>515.3</td>
<td>34.7</td>
</tr>
<tr>
<td>s13207</td>
<td>11280</td>
<td>59.9</td>
<td>262.8</td>
<td>9.4</td>
<td>102.5</td>
<td>24.4</td>
<td>443.4</td>
<td>29.8</td>
</tr>
<tr>
<td>s9234</td>
<td>7777</td>
<td>176.2</td>
<td>201.8</td>
<td>7.3</td>
<td>82.9</td>
<td>19.9</td>
<td>318.9</td>
<td>22.6</td>
</tr>
<tr>
<td>s5378</td>
<td>3993</td>
<td>188.9</td>
<td>121.9</td>
<td>4.8</td>
<td>41.1</td>
<td>9.7</td>
<td>250.3</td>
<td>17.3</td>
</tr>
<tr>
<td>s1494</td>
<td>1552</td>
<td>427.5</td>
<td>180.4</td>
<td>5.9</td>
<td>19.4</td>
<td>4.8</td>
<td>90.3</td>
<td>6.5</td>
</tr>
<tr>
<td>s713</td>
<td>628</td>
<td>402.1</td>
<td>219.9</td>
<td>9.1</td>
<td>7.8</td>
<td>2.0</td>
<td>36.6</td>
<td>2.6</td>
</tr>
<tr>
<td>s641</td>
<td>594</td>
<td>581.4</td>
<td>205.1</td>
<td>8.8</td>
<td>6.8</td>
<td>1.8</td>
<td>33.5</td>
<td>2.5</td>
</tr>
<tr>
<td>s298</td>
<td>264</td>
<td>616.8</td>
<td>33.3</td>
<td>1.6</td>
<td>2.8</td>
<td>0.7</td>
<td>12.4</td>
<td>0.9</td>
</tr>
<tr>
<td>s27</td>
<td>16</td>
<td>205.0</td>
<td>25.4</td>
<td>1.2</td>
<td>0.2</td>
<td>0.1</td>
<td>1.3</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Figure 3.11: STD errors of $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$ for s38584 ($T_{op} = 298K$)

These macromodels characterize the delay and power consumption of logic gates and memory cells under PVT variations and at various temperatures. We illustrated the construction flow of the FinFET design library. We then used these macromodels to develop statistical delay and power analysis techniques. Using the rectangular grid-based method, we were able to model the spatial correlations of the PV parameters. We used QMC simulations to characterize the effects of PV variations on delay and leakage of logic gates, and identified $L_G$, $T_{SI}$, and $V_{DD}$ as the main parameters that affect gate delay, and $L_G$, $T_{SI}$, and $\Delta \Phi_{GN}$ as the main parameters that affect leakage power. We investigated the impact of PVT variations on the delay and power of logic gates/circuits and memory cells. We validated the PVT variation models by comparing them with QMC simulation results. The QMC test showed that the proposed delay model as compared to the QMC results had an average statistical error of only 3.4% and 4.4% for $S_{out}$ and $d_{gate}$, respectively. For the leakage power model, on an average, the statistical error was only 3.1%. We compared FinFET logic gates implemented in various design styles, and showed that ASG-mode logic gates can provide better performance in terms of both delay and power than IG-mode logic gates. We explored four different designs of memory cells,
including 4T [60], PGFB 6T [62], RBGB 6T [42], and 8T SRAM cells [61], and found that RBGB has the lowest leakage power as compared to the other three types of cells. Finally, we validated our PVT variation models at the circuit level against the MC simulation results, and showed the mean and STD of $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$, respectively. The errors of the evaluated STDs with respect to those obtained from the MC simulations are, in general, less than 5.0%, 1.0%, and 1.0% for the STDs of $D_{Ckt}$, $P_{Leak}$, and $P_{Dyn}$, respectively.
Chapter 4

FinCANON Simulation Framework

4.1 Introduction

FinCANON \cite{FinCANON} is an integrated framework that simulates power, delay, as well as PVT variations of caches and on-chip networks. Fig. 4.1 shows the block diagram of FinCANON. Its key components are: (i) FinFET design library that contains the circuit-level parameters of logic gates, memory cells, and voltage generators, and characterizes the effects of PVT variations, (ii) FinCANON architectural model for power and delay analysis, (iii) integration into GEM5 \cite{GEM5}, the CMP simulator for cycle-accurate performance simulations, and (iv) CMP representation to provide necessary parameters to the FinFET architectural model. FinCANON is modular, with the cache and NoC components, models, and FinFET design library programmed in a hierarchical, object-oriented fashion, enabling them to be easily updated later.

The FinCANON architectural model receives inputs from the CMP representation, and generates power/area/timing profiles based on the FinFET design library. The FinCANON architectural model consists of two major parts: CACTI-PVT for caches and memories, and ORION-PVT for NoCs. Both parts are capable of modeling PVT variations based on the specified extent of variations. FinCANON generates the
power profile in collaboration with GEM5 [125], in which GARNET-FinFET [70] and Ruby [125] model the runtime statistics of on-chip networks and caches, respectively. Based on these statistics, CACTI-PVT and ORION-PVT generate the power profile for the target system.

The CMP representation block specifies system structures and environment parameters that need to be simulated. It consists of three major parts: CMP architecture configuration, environment parameters, and DPM/DTM schemes. The CMP architecture configuration specifies the CMP structure (e.g., number of cores, network configuration, cache size, etc.). The environment parameters contain information about the IC environment (e.g., PVT variations, supply voltages, temperature, etc.). The CMP representation block also allows DPM/DTM schemes to be defined.

Figure 4.1: FinCANON modeling framework
4.1.1 Architecture-level macromodels

FinCANON characterizes cache and NoC components with macromodels. The macromodel approach has been used extensively in simulators such as CACTI [100], ORION [101], Wattch [110], and McPAT [33]. However, they consider nominal values only, and do not take PVT variations into consideration. In this dissertation, the macromodels have been enhanced to include delay variations [98, 138, 130] and power variations [149]. The macromodel of a component contains delay/power information, and their STDs. These macromodels are implemented in a hierarchical fashion. Fig. 4.2 shows the hierarchy of the macromodels. The hierarchy spans the highest architecture level to the lowest logic level. The delay, power, and PVT variations of a component are derived from its lower-level macromodels. For example, the macromodel of a cache consists of the macromodels of decoders, subarrays, data
buses, etc. At the lowest FinFET level, the macromodel of a FinFET logic gate is directly obtained from the FinFET design library.

The macromodel methodology facilitates calculation of spatial correlations among architectural components, allowing FinCANON to estimate the architectural impact of PVT variations. In a CMP with millions of gates, calculating the spatial correlations for all pairs of gates requires an impractical amount of CPU time. As the gate count increases, the required CPU time grows exponentially. Calculating the spatial correlations among components, however, requires significantly less time [149]. Therefore, FinCANON calculates the spatial correlations between pairs of logic gates only within each component, and calculates the spatial correlations between pairs of components at the architecture level. For instance, it calculates correlations between memory cells to generate the subarray macromodel, and correlations of subarrays in a bank to generate the bank macromodel. The macromodel methodology allows different components to have different amounts of PVT variations. This feature makes the macromodel methodology compatible with the rectangular grid-based method. Another benefit of using macromodels is the ease with which various architectures can be modeled. New components can be added to FinCANON’s framework by simply inserting new macromodels into it. Existing components can be changed by modifying their macromodels.

4.2 CACTI-PVT

Fig. 4.3 shows the block diagram of the CACTI-PVT modeling framework. CACTI-PVT is enhanced from CACTI-FinFET [99]. The improvements in CACTI-PVT are as follows.

- Its FinFET design library is based on 2D TCAD FinFET models that capture the simulation accuracy of 3D TCAD models [73]. In CACTI-FinFET, the
FinFET design library was generated using the University of Florida double-gate (UFDG) model [162], which is a process/physics-based double-gate FET model. UFDG is not always able to model FinFETs at the 22nm node due to convergence problems.

- In addition to process variations, CACTI-PVT enables modeling of voltage and temperature variations.

- The leakage power of each cache component in CACTI-PVT is modeled as a lognormal RV. In CACTI-FinFET, it is modeled as an RV with a normal distribution.
• The delay variation model in CACTI-PVT has been updated with a modified Horowitz delay approximation [88], as described in the previous chapter.

• The memory cell models are calibrated based on accurate 2D TCAD simulations.

• It supports ASG-mode FinFETs.

CACTI-PVT is divided into four key components: (i) FinFET cache model that calculates the nominal values of power and delay of the cache, (ii) PVT variation model for generating delay and leakage spreads, (iii) cache configurations to provide system-level cache design parameters, and (iv) spatial grid assignment block to specify the PVT variations in different locations of a die. A number of circuit-level improvements have been made to CACTI’s cache model. The cache structure block determines the structures of the cache components. The PVT variation parameters block determines the PVT parameters to be simulated in the PVT variation model. The cell configurations block specifies the design styles of FinFET logic gates and memory cells.

The outputs of CACTI-PVT consist of delay and power profiles. The delay profile contains a summary of the search delay of the tag array, the write/read delay of the data array, as well as the corresponding delay spread information. The power profile provides the write/read energy per access, and the dynamic/leakage power breakdown of the entire cache. The nominal values of delay and power are provided by the cache model, while their spread information is obtained from the PVT variation model. CACTI-PVT is then able to derive distributions of the overall power and delay of FinFET-based caches.
4.2.1 Mat organization

Fig. 4.4 shows the mat organization in a FinFET cache. A mat is a basic building block in CACTI [100]. It contains four subarrays, pre-decoder, decoder, pre-charge circuitry, and drivers. CACTI-PVT uses the same structure. In case memory cells need to be independently biased, a voltage generator (VG) is included in the mat. Bias lines are distributed across the subarray to drive the back gates of the relevant FinFET memory cells. If multi-operation modes are needed (e.g., active mode, low power mode, etc.), a mode controller (MC) is added to the mat. A bias voltage decoder and mode-control signal decoder are added when a voltage generator and MC are included.

The FinFET design library supports several memory cell styles including 4T [60], PGFB 6T [62], RBGB 6T [42], and 8T SRAM cells [61]. When a specific cell style is invoked, its characteristics are fed to the cache model. The corresponding peripheral components are also changed to meet the cell structure requirement. This modular library enables researchers to easily add new FinFET memory cell styles.

4.3 ORION-PVT

ORION-FinFET [89] is a FinFET-based power simulator for NoCs. It was built on top of the ORION infrastructure that targets bulk CMOS. ORION-FinFET enables users to explore router power consumption under two FinFET design styles and various temperatures. It includes NoC power models and a FinFET power library. The power models describe how the power of router components is calculated from the capacitance and leakage current of logic gates. The power library specifies the capacitance and leakage current values of logic gates for different FinFET design styles and at different temperatures. For instance, it contains the capacitance and leakage current from a minimum-sized inverter used in the arbiter to the large inverters used
Figure 4.4: Hierarchical organization of a mat
in a link driver. Given the fin count, the capacitance and leakage current of logic gates are linearly scaled from those of the minimum-sized logic gates. The calculated capacitance values can be directly used with ORION’s capacitance equations. ORION-FinFET’s power models and power library, however, do not model PVT variations and access delay of router components. It has been shown that the delay uncertainty caused by PVT variations is becoming an increasing fraction of the clock period [20]. In addition, it has been pointed out [94] that when PVT variations exist, each router in an NoC must synchronize with the slowest router, leading to degradation in the throughput of the entire NoC. Therefore, to build a FinFET-based NoC simulator for the deep-submicron technology nodes, several further revisions to ORION-FinFET are necessary.

ORION-PVT is an extension of ORION-FinFET. Fig. 4.5 shows the block diagram of the ORION-PVT modeling framework. ORION-PVT contains four major revisions that make it different from the previous version: FinFET design library, delay models, new cache models, and PVT variation models. The previous version of ORION-FinFET is based on a FinFET power library for the 32nm node. This power library was generated using the UFDG model. ORION-PVT replaces the FinFET power library with the FinFET design library, which is characterized by TCAD device simulation that provides higher accuracy. It incorporates delay models for the router components, so that critical path delay can be estimated for different router pipeline configurations. Moreover, it replaces ORION-FinFET’s original cache model, which was based on an older version of CACTI, with CACTI-PVT’s cache model. ORION-PVT, like CACTI-PVT, supports PVT variation modeling capability, so that the impact of PVT variations on performance and power consumption of the entire NoC can be estimated.
4.3.1 Delay and power models of NoC components

ORION-PVT’s router model is enhanced from ORION 2.0’s model for bulk CMOS. ORION-PVT has modular models for the router components, which are all implemented in an object-oriented fashion. Following FinCANON’s methodology, the router components are characterized through macromodels. The macromodels fall into the following three categories:

1. On-chip router: buffers, crossbar, arbiters, and DFF arrays.

2. Clock tree: Local and global clock trees, drivers, clock wires, and loads.

3. Interrouter links: link wires and drivers.
The input buffers in a router are directly modeled by CACTI-PVT. Note that in practical CMPs, such as Tilera \[7, 8\], cache-based input buffers are not used because the required buffer sizes are typically small. ORION-PVT has an interface with CACTI-PVT and can, hence, model any cache-like component by a function call to CACTI-PVT, as shown in Fig. 4.5. In ORION-PVT, the input buffers can be implemented with any of the four memory cell styles supported by CACTI-PVT. The buffer model used in the original ORION/ORION-FinFET is obsolete and has been removed. The peripheral circuitry of the input buffer can be implemented using any FinFET design style. This enhancement enables flexible modeling of the router architecture. If additional buffers are added to the router design, ORION-PVT can easily model the overhead by initiating additional instances of CACTI-PVT’s cache modules.

In addition to the input buffers, DFF arrays are inserted into the router at the following interfaces: interrouter links/input buffers, input buffers/crossbar, and crossbar/interrouter links. The width of the DFF array at these locations is equal to the flit width. These arrays dissipate both dynamic and leakage power, and their capacitance contributes to the load capacitance $C_{ClkLoad}$ of the clock tree. In terms of dynamic power, DFF arrays share the same traffic load value as the input buffer. The power dissipated by DFF arrays is incorporated into the buffer power consumption.

The models of the crossbar and arbiter are enhanced as well. The previous model calculates their power consumption by making functional calls. In ORION-PVT, they are both implemented using macromodels. In addition to power consumption, these macromodels calculate critical path delay and impact of PVT variations as well. This information is critically important in the design phase, since the crossbar usually has the longest delay in a router.

Both ORION-FinFET and ORION-PVT contain clock/link models, which are not supported in ORION (however, ORION 2.0 includes such models). We use clock/link
models given in [163, 164]. An H-tree is adopted as the clock tree structure (Intel’s
teraflops processor chip [5] employs an H-tree). Clock drivers are distributed across
the entire H-tree, rather than placing a single large driver at the root. The distributed
driver scheme is much more power-efficient than the single driver scheme [164]. Clock
drivers are inserted at the root of each branch of the clock tree. Driver sizes are
designed such that the delay of each branch is equal to an inverter fanout-of-four
(FO4) delay. The clock power $P_{Clk}$ of a router can be calculated as:

$$P_{Clk} = f_{Clk} \times (C_{H-tree} + C_{ClkLoad} + C_{ClkDriver}) \times V_{DD}^2$$ (4.1)

where $C_{H-tree}$ is the wire capacitance of the H-tree, $C_{ClkLoad}$ the total load capaci-
tance, and $C_{ClkDriver}$ the sum of the total driver capacitance in the H-tree. $C_{H-tree}$ is
composed of several levels. For a four-level H-tree, $C_{H-tree}$ can be computed as:

$$C_{H-tree} = C_U \times D_R \times [8 \times 1/4 + 4 \times 1/4 + 2 \times 1/2 + 1 \times 1/2]$$ (4.2)

where $C_U$ is the global wire capacitance per unit length, and $D_R$ the dimension of
the router. Each term in (4.2) represents (number of segments per level) $\times$ (fraction
of $D_R$ per segment at that level).

The link power $P_{Link}$ depends on the flit width and the number of links in the
network. It can be calculated as:

$$P_{Link} = 0.5 \times \alpha \times f_{Clk} \times (C_U \times L + C_{LinkLoad} + C_{LinkDriver}) \times V_{DD}^2 \times F \times N_{Link}$$ (4.3)

where $F$ is the flit width, $C_{LinkLoad}$ the total load capacitance, $C_{LinkDriver}$ the sum of
the total driver capacitance of a single link, $N_{Link}$ the total number of the links in the
network, and $L$ the length of a single link. Note that the leakage power dissipated by
the clock net and links is only contributed by the drivers in them.
The whole network power is obtained by summing the router power, clock power, and link power, which can be summarized using the equation:

\[ P_{\text{Network}} = N_{\text{Tile}} \times (P_{\text{Buffer}} + P_{\text{Crossbar}} + P_{\text{Arbiter}}) + P_{\text{Link}} + P_{\text{Clk}} + P_{\text{NetLeak}} \]  

(4.4)

where \( N_{\text{Tile}} \) is the total number of tiles in the network, \( P_{\text{Buffer}} \) the power of the buffers, \( P_{\text{Crossbar}} \) the power of the crossbar, \( P_{\text{Arbiter}} \) the power of the arbiters, and \( P_{\text{NetLeak}} \) the total leakage power in the interconnection network.

### 4.4 GARNET-FinFET

We have developed GARNET-FinFET, which is based on GARNET [126], to model the power and performance of a FinFET-based NoC. GARNET is a cycle-accurate performance simulator for interconnection networks, and incorporates the original ORION router power model [31]. Hence, it provides a complete simulation framework to simulate the power consumption based on the traffic in the network.

Fig. 4.6 illustrates how GARNET-FinFET works. The network topology is first specified by the user. GARNET-FinFET then injects flits into the network based on the given topology. When the update period is reached, it generates router utilization statistics of the previous period and calls ORION-PVT to do power simulation. ORION-PVT then uses the statistics to simulate the power consumption based on the FinFET design library. After power simulation is done, GARNET-FinFET checks if the overall simulation is completed. If so, GARNET-FinFET reports the network power. Otherwise, it returns to traffic simulation for the next update period and goes through the loop again. It calculates the average network latency as well, thereby allowing us to compare the performance for different traffic patterns and router configurations.
4.5 Experimental Results

In this section, we report results for FinCANON simulations of FinFET-based caches and NoCs for a number of cache and NoC configurations, FinFET logic styles (SG, IG, and ASG modes), supply voltages, and temperatures. Table 4.1 shows the default cache configuration parameters assumed in our experiments. The cache capacity is assumed to be 4MB. Each subarray contains 64 wordlines and 64 bitlines. The wordline and bitline lengths are 16.9\(\mu m\) and 16.9\(\mu m\), respectively. If not specifically stated, the operating temperature is assumed to be 298K. By default, the cache implementation is assumed to have SG-mode peripheral components combined with PGFB cells. The PV parameters considered in the experiments include \(L_G, T_{SI}, T_{OX}, \Delta \Phi_{GN}, \Delta \Phi_{GP},\) and \(V_{DD}\). The nominal values of the PVT parameters are listed in Table 4.1. In all cases, we assume a normal distribution of PV parameters and set
Table 4.1: Cache configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity (MB)</td>
<td>4</td>
</tr>
<tr>
<td>Block size (bytes)</td>
<td>64</td>
</tr>
<tr>
<td>Associativity</td>
<td>4</td>
</tr>
<tr>
<td>Cache access model</td>
<td>uniform cache architecture</td>
</tr>
<tr>
<td>Number of read/write ports</td>
<td>1</td>
</tr>
<tr>
<td>Number of banks</td>
<td>4</td>
</tr>
<tr>
<td>Subarray dimensions</td>
<td>64 bits (horizontal) \times 64 bits (vertical)</td>
</tr>
<tr>
<td>Wordline length</td>
<td>16.9\mu m</td>
</tr>
<tr>
<td>Bitline length</td>
<td>16.9\mu m</td>
</tr>
<tr>
<td>Output width (bits)</td>
<td>512</td>
</tr>
<tr>
<td>Technology</td>
<td>22nm FinFET</td>
</tr>
<tr>
<td>$V_{DD}$ (V)</td>
<td>0.9</td>
</tr>
<tr>
<td>Varying PV parameters</td>
<td>$L_g$, $T_{Si}$, $T_{OX}$, $\Delta\Phi_{GN}$, $\Delta\Phi_{GP}$, and $V_{DD}$</td>
</tr>
<tr>
<td>Temperature (K)</td>
<td>298</td>
</tr>
<tr>
<td>$3\sigma/\mu$</td>
<td>10%</td>
</tr>
</tbody>
</table>

Table 4.2: Network parameters

<table>
<thead>
<tr>
<th>Network parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>$1GHz$</td>
</tr>
<tr>
<td>$V_{DD}$ (V)</td>
<td>0.9</td>
</tr>
<tr>
<td>Technology</td>
<td>22nm FinFET</td>
</tr>
<tr>
<td>Topology</td>
<td>4\times4 Mesh</td>
</tr>
<tr>
<td>Number of message classes</td>
<td>4</td>
</tr>
<tr>
<td>Link length</td>
<td>$1mm$</td>
</tr>
<tr>
<td>Router ports</td>
<td>3~5</td>
</tr>
<tr>
<td>Input buffer IO ports</td>
<td>1 read, 1 write</td>
</tr>
<tr>
<td>Flit size</td>
<td>$128$ bits</td>
</tr>
<tr>
<td>VCs per message class</td>
<td>8</td>
</tr>
<tr>
<td>Buffers per port</td>
<td>48 flits (dynamically allocated)</td>
</tr>
<tr>
<td>Link latency</td>
<td>1 cycle</td>
</tr>
<tr>
<td>Routing algorithm</td>
<td>Dimension-ordered X-Y</td>
</tr>
</tbody>
</table>

the $3\sigma$ value of the distribution to 10% of the nominal value. The value of $\sigma$ is evenly distributed to three levels of grids ($\sigma_{Bank} = \sigma_{Array} = \sigma_{Subarray} = \sigma/\sqrt{3}$).

Table 4.2 shows the network parameters along with the routing algorithm used. These numbers are fed to GEM5. Each packet consists of four flits. The number of router ports ranges from three to five because the routers in the corner and along the
Table 4.3: Variable PVT parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Nominal value</th>
<th>Range</th>
<th>Step Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate length, $L_G$ (nm)</td>
<td>20</td>
<td>[18,22]</td>
<td>0.4</td>
</tr>
<tr>
<td>Silicon thickness, $T_{SI}$ (nm)</td>
<td>10</td>
<td>[9,11]</td>
<td>0.2</td>
</tr>
<tr>
<td>Oxide thickness, $T_{OX}$ (nm)</td>
<td>1</td>
<td>[0.9,1.1]</td>
<td>0.02</td>
</tr>
<tr>
<td>nFinFET workfunction, $\Phi_N$ (eV)</td>
<td>4.4</td>
<td>[4.38,4.42]</td>
<td>0.004</td>
</tr>
<tr>
<td>pFinFET workfunction, $\Phi_P$ (eV)</td>
<td>4.8</td>
<td>[4.78,4.82]</td>
<td>0.004</td>
</tr>
<tr>
<td>Supply voltage, $V_{DD}$ (V)</td>
<td>0.9</td>
<td>[0.8,1.0]</td>
<td>0.02</td>
</tr>
<tr>
<td>Temperature, $T_{op}$ (K)</td>
<td>N.A.</td>
<td>[298,398]</td>
<td>10</td>
</tr>
</tbody>
</table>

sides have fewer input ports than the routers in the middle of the mesh. The number of simulation cycles is 1,000,000 and includes 500,000 cycles for warm-up. By default, the router is assumed to have SG-mode components combined with PGFB cells. The packet injection rate is assumed to be 0.04 packets/node/cycle. Both synthetic and real traffic patterns are used in our simulations. PARSEC is used to obtain real traffic patterns [103]. It is a benchmark suite composed of multithreaded programs. It is specifically designed to represent emerging workloads of next-generation shared-memory programs running on CMPs.

4.5.1 Cache delay and leakage characterization

Fig. 4.7 shows the delay and leakage STDs of a cache with $L_G$, $T_{SI}$, $T_{OX}$, $\Delta \Phi_{GN}$, $\Delta \Phi_{GP}$, and $V_{DD}$ individually varying normally. The nominal values and variation ranges of these parameters are shown in Table 4.3. The nominal values of delay and leakage power are 1.83ns and 285.20mW, respectively, which are the same for all cases. Fig. 4.7(a) shows the delay STDs versus the PV parameters. The delay STD is more pronounced in the $L_G$ case. The cache delay is more sensitive to $T_{SI}$ variation than $T_{OX}$ variation. The delay STD is not sensitive to $\Delta \Phi_{GN}$, $\Delta \Phi_{GP}$, and $V_{DD}$ variations. Fig. 4.7(b) shows the leakage power STDs versus the PV parameters.
(a) Cache delay STD vs. PV parameters

(b) Cache leakage power STD vs. PV parameters

Figure 4.7: Cache delay and leakage characterization
Figure 4.8: Cache delay and leakage vs. $3\sigma/\mu$ variations

The leakage power is more pronounced in the $L_G$, $T_{SL}$, and $\Delta \Phi_{GN}$ cases. Variations in $T_{OX}$, $\Delta \Phi_{GP}$, and $V_{DD}$ have little impact on the cache leakage power.

### 4.5.2 Cache delay and leakage vs. $3\sigma/\mu$ variations

Fig. 4.8 compares the delay and leakage STDs of the cache for various $3\sigma/\mu$ values of the PV parameters. Variations in all the PV parameters are enabled. The nominal values of delay and leakage power are the same as those in the previous subsection. It is observed that as $3\sigma/\mu$ increases, both the delay and leakage STDs increase linearly.
The delay STD trend conforms to (3.10), while the leakage STD trend matches the equations in [149].

### 4.5.3 Delay and leakage vs. temperature

Fig. 4.9 plots the cache delay, leakage power, and their STDs as a function of temperature. Variations in $L_G$, $T_{SI}$, $T_{OX}$, $\Delta\Phi_{GN}$, $\Delta\Phi_{GP}$, and $V_{DD}$ are all enabled. The memory cell type is PGFB. Fig. 4.9(a) shows the cache delay trend where SG-mode cache peripheral circuitry is used. As the temperature increases from 298K to 378K, the nominal delay and its STD increase by 8.7% and 5.0%, respectively. CMOS-based caches show a much higher access time increase of as much as $2.03 \times$ at high temperatures due to mobility degradation [146]. Fig. 4.9(b) compares cache delay when the three FinFET design styles are used for the cache peripheral circuitry. The IG-mode delay grows faster than SG-/ASG-mode delays as the temperature increases from 298K to 378K. The ratios of the IG-mode delay to ASG-mode delay at 298K and 378K are $1.19 \times$ and $1.24 \times$, respectively. Fig. 4.9(c) shows the cache leakage trend for the SG-mode case. The cache leakage power and its STD both increase exponentially. The leakage power and its STD at 378K are $19.45 \times$ and $19.25 \times$, respectively, as compared to those at 298K.

### 4.5.4 Comparison of FinFET operation modes

Fig. 4.10 compares the cache delay and leakage power for various combinations of memory peripheral components and memory cells. The peripheral components include the input/output H-tree of the datapath, decoders, wordline/bitline drivers, and precharge circuitry. For example, SG+PGFB corresponds to SG-mode peripheral components combined with PGFB cells. IG- and ASG-mode logic gates are also used to implement the peripheral components and compared against the SG-mode implementation. The IG-mode gates are biased at $V_{HI} = 1.1V$ and $V_{LOW} = -0.2V$. 

87
Figure 4.9: Cache delay and leakage vs. temperature
Figure 4.10: Comparison of FinFET operation modes
Two memory cell styles are compared: PGFB and RBGB. The RBGB memory cells are biased at $-0.2\,\text{V}$. From Fig. 4.10(a) we can see that the nominal delays of the IG-mode implementations are larger than those of the SG- and ASG-mode implementations. Changing the memory cell style from PGFB to RBGB does not have a significant impact on cache delay. The reason is twofold. First, the PGFB cell has the same layout area as the RBGB cell, as shown in Fig. 3.6(b). Second, peripheral components are the main contributors to cache delay, not the memory cells. For the IG-mode implementations, the nominal delay as compared to the SG-mode and ASG-mode implementations is $1.27\times$ and $1.19\times$, respectively. The delay of the ASG-mode implementations is 6.6% larger than that of the SG-mode implementations.

Fig. 4.10(b) compares the cache leakage power for the different implementations. We can see that when RBGB cells are used, the leakage power is reduced significantly. When SG-mode peripheral components are used, the leakage power of the PGFB implementation is $3.31\times$ compared to that of the RBGB implementation. The SG-mode peripheral components account for 30.2% of the leakage power. When PGFB cells are used, the IG- and ASG-mode peripheral components reduce the total leakage power by 23.9% and 25.3%, respectively, compared to the SG-mode implementation. When RBGB cells and IG-/ASG-mode peripheral components are used together, the total leakage power is reduced by 93.6% and 95.2%, respectively, compared to the SG+PGFB implementation.

From the results in Fig. 4.10, we conclude that IG-mode peripheral components, when combined with PGFB cells, provide little help in reducing cache leakage power, but increase cache delay considerably. On the other hand, RBGB cells reduce leakage power significantly, with smaller impact on cache access delay. Therefore, it is advisable to reverse-bias the memory cells to achieve the most leakage power reduction, and use SG-mode logic gates in the peripheral components to obtain the fastest cache access time. If leakage power is the most important concern, then use of ASG-mode
peripheral components plus RBGB cells is advisable, with only a slight increase in cache delay.

### 4.5.5 Delay and leakage vs. cache capacity

Fig. 4.11 compares the delay and leakage power for various cache capacities of the SG+PGFB implementation. Delay, leakage, and their STDs all increase as the cache capacity increases, as expected. For large-capacity caches, increased delay variation implies that balancing the access times in different banks/mats becomes more difficult. Since smaller banks have smaller delay variations, partitioning large caches into smaller banks is a promising way to overcome process variations. Cache partitioning does not lead to smaller variations in the cache leakage power because it is the sum of the leakage power of all the banks. Fig. 4.11(c) shows the cache leakage per bit, which converges to 8.74nW as the cache size increases. When the cache size is small, the cache leakage per bit is affected by peripheral circuitry. When the cache size grows, the cache leakage per bit is dominated by memory cells.

### 4.5.6 Delay and leakage vs. memory styles

Fig. 4.12 compares the cache performance for different types of memory cells. The memory peripheral components are all implemented in the SG mode. We can see from Fig. 4.12(a) that the nominal delay and STD grow with the number of transistors in the memory cell. More transistors in a memory cell imply a larger cache area and, hence, lead to larger address/data H-trees and longer wordlines/bitlines, thus increasing the critical path delay and its STD. The nominal delay of the RBGB, 4T SRAM, and 8T SRAM implementations are 1.00×, 0.98×, and 1.25×, respectively, compared to the PGFB implementation. Fig. 4.12(b) shows the cache leakage power comparison. The nominal leakage power of the RBGB, 4T SRAM, and 8T SRAM implementations are 0.30×, 0.64×, and 1.82×, respectively, compared to the PGFB
Figure 4.11: Cache delay and leakage vs. capacity for the SG+PGFB configuration
Figure 4.12: Comparison of different memory styles (peripheral components in SG mode)
implementation. The nominal leakage power and its STD also grow with the number of transistors in the memory cell, except for the IG-mode RBGB 6T SRAM cell case. This is because its leakage power is primarily governed by the reverse bias voltage applied. It appears that the 4T SRAM cell performs much better than the 8T SRAM cell from both the delay and power points of view. However, when stability and reliability are taken into account, 8T SRAM cells are much superior to 4T SRAM cells. This is because the read operation does not discharge current from the cells. During a read operation, the 8T SRAM cell is connected to the gate of a pass transistor in the bitline discharging path \[61\]. The memory cell itself thus is not discharged. On the other hand, the contents of 4T SRAM cells are vulnerable to read operations \[60\]. 4T SRAM cells do not have pull-up PMOS transistors, thus have an even smaller read margin than 6T SRAM cells \[60\].

4.5.7 Delay and power of router components

Tables 4.4 and 4.5 show delay, leakage power, and their STDs of router components. The last three columns correspond to the three FinFET logic styles used to implement the router components. Input buffers are assumed to have PGFB cells. The network parameters are summarized in Table 4.2. The network topology is 4×4 mesh. The operating frequency and temperature are 1GHz and 298K, respectively. If not specifically stated, the default network parameters are assumed to be these values. Among the three router components, the crossbar has the longest delay. On the other hand, the input buffers are the main contributors to leakage power. When IG-mode logic gates are used for the router components, the nominal delay and STD increase significantly relative to the SG-mode case. The IG- and ASG-mode crossbars have 128.4% and 29.4% longer delay than the SG-mode crossbar. The IG- and ASG-mode crossbars have 88.1% and 98.2% lower leakage power, respectively, relative to the SG-mode crossbar. The IG- and ASG-mode arbiters have 88.6% and 97.4% lower
Table 4.4: Delay of router components

<table>
<thead>
<tr>
<th>Router component</th>
<th>Delay</th>
<th>SG</th>
<th>IG</th>
<th>ASG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input buffer</td>
<td>Nominal value (ns)</td>
<td>0.15</td>
<td>0.37</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>STD (ps)</td>
<td>3.18</td>
<td>8.62</td>
<td>2.36</td>
</tr>
<tr>
<td>Crossbar</td>
<td>Nominal value (ns)</td>
<td>1.97</td>
<td>4.50</td>
<td>2.55</td>
</tr>
<tr>
<td></td>
<td>STD (ps)</td>
<td>29.60</td>
<td>147.30</td>
<td>29.02</td>
</tr>
<tr>
<td>Arbiter</td>
<td>Nominal value (ns)</td>
<td>0.22</td>
<td>0.41</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>STD (ps)</td>
<td>1.65</td>
<td>3.46</td>
<td>1.50</td>
</tr>
</tbody>
</table>

Table 4.5: Leakage power of router components

<table>
<thead>
<tr>
<th>Router component</th>
<th>Leakage</th>
<th>SG</th>
<th>IG</th>
<th>ASG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input buffer</td>
<td>Nominal value (µW)</td>
<td>5541.09</td>
<td>5474.31</td>
<td>5466.52</td>
</tr>
<tr>
<td></td>
<td>STD (µW)</td>
<td>1401.04</td>
<td>1380.15</td>
<td>1377.81</td>
</tr>
<tr>
<td>Crossbar</td>
<td>Nominal value (µW)</td>
<td>268.84</td>
<td>31.95</td>
<td>4.87</td>
</tr>
<tr>
<td></td>
<td>STD (µW)</td>
<td>46.53</td>
<td>5.04</td>
<td>0.27</td>
</tr>
<tr>
<td>Arbiter</td>
<td>Nominal value (µW)</td>
<td>2216.80</td>
<td>252.04</td>
<td>57.57</td>
</tr>
<tr>
<td></td>
<td>STD (µW)</td>
<td>328.33</td>
<td>32.57</td>
<td>2.12</td>
</tr>
</tbody>
</table>

leakage power, respectively, relative to the SG-mode arbiters. IG- and ASG-mode implementations reduce the leakage power of the input buffers by only 1.2% and 1.4%, respectively, relative to the SG-mode input buffers. This is because the leakage power of the input buffers is mainly contributed by the PGFB cells.

4.5.8 Network power breakdown

Fig. 4.13 shows the network power for four different router configurations. The operating temperature is assumed to be 298K. The dynamic power in the SG+PGFB, IG+PGFB, ASG+PGFB, and ASG+RBGB implementations are 56.4%, 62.8%, 66.0%, and 95.9% of the total power, respectively. The IG+PGFB implementation has less dynamic power consumption than other implementations since only one of the two gates in the IG-mode FinFETs is used for logic operation. The SG+PGFB implementation has the highest leakage power consumption since both the memory
and logic cells leak significantly. The ASG+RBGB implementation minimizes leakage power consumption in both memory and logic cells.

Fig. 4.14 gives the power breakdown of the entire network for the four cases shown in Fig. 4.13. Dynamic and leakage power are presented separately for a clear comparison. Dynamic power includes contributions from the input buffers, crossbar, arbiters, and clock tree. In all cases, the dynamic power is mainly contributed by the crossbar, which consists of drivers and long wires, corresponding to large load capacitances. In Fig. 4.14(a), the leakage power from the input buffers contribute 28% of the total power. The arbiters contribute 14% of the total power. In Figs. 4.14(b) and 4.14(c), the leakage power from the arbiters is reduced to 2% and less than 1%, respectively. In Fig. 4.14(d), the input buffer leakage power shrinks to only 4% due to the use of RBGB cells. It can be seen that almost all of the power consumption comes from dynamic power in this case.

4.5.9 Power comparison for various types of traffic patterns

Fig. 4.15 compares the power consumption of SG-, IG-, and ASG-mode routers for three kinds of traffic patterns. These traffic patterns include uniform random, tor-
Figure 4.14: Pie charts of network power

(a) SG-mode network power breakdown with PGFB cells
(b) IG-mode network power breakdown with PGFB cells
(c) ASG-mode network power breakdown with PGFB cells
(d) ASG-mode network power breakdown with RBGB cells

nado, and bit-complement traffic [126]. In the uniform random traffic pattern, as the name suggests, packets are sent to destinations in a uniform random fashion. The bit-complement traffic involves transmissions of packets from nodes to their bit-complement nodes. In tornado traffic, packets are passed from router to router in a tornado-like manner. Different traffic patterns have different link and router port utilization rates, thus dissipate different amounts of dynamic power. The packet injection rate for the results in Fig. 4.15 is assumed to be 0.04 packets/node/cycle. We can see that the power reduction due to the use of the IG and ASG modes is similar for all traffic patterns. This is because leakage power is not affected by the network traffic applied. Therefore, power reduction with the IG or ASG modes is more significant when the network has lighter traffic. When the traffic is uniform random, the
IG- and ASG-mode implementations have 20.4% and 17.2% lower power, respectively, than the SG-mode implementation.

### 4.5.10 Network power vs. temperature

Fig. 4.16 plots network power for the ASG+PGFB case as a function of temperature. The network leakage power grows exponentially as the temperature increases. At 378K, the leakage power increases by $20.83 \times$ relative to that at 298K. The dynamic power is not affected significantly by temperature. As a result, leakage power as a fraction of total power consumption increases as the temperature increases.

### 4.5.11 PARSEC simulations

We use GARNET-FinFET to perform real-traffic simulations on a 16-core CMP. Cores have private 64KB L1 instruction and 64KB L1 data caches and share a distributed 4MB L2 cache. The implementation is SG+PGFB. The operating temperature is assumed to be 378K. The caches are kept coherent using a MOESI directory-based protocol. Each core is connected to a $4 \times 4$ mesh with dimension-ordered routing. Nine PARSEC benchmarks [103] are included in the simulation. Only critical
regions of the benchmarks are simulated. Fig. 4.17(a) shows the network latency. The average network latency is 25.02 cycles. However, these traffic loads are low, in general, compared to synthetic traffic. This is because PARSEC benchmarks do not inject traffic that is as heavy as in the synthetic cases. Fig. 4.17(b) shows the network power consumption. It can be seen that leakage power constitutes most of the power consumption even in real-traffic simulations. On an average, 77.4% of the power consumption comes from leakage power.

4.6 Chapter Summary

In this chapter, we presented FinCANON, which enables modeling of delay, power, and PVT variations for FinFET-based caches and NoCs. It consists of CACTI-PVT and ORION-PVT for modeling caches and NoCs, respectively. It is built atop a FinFET design library. It makes use of gate-level macromodels in this library at higher levels of the design hierarchy. We described the modeling frameworks of both CACTI-PVT and ORION-PVT. We illustrated the impact of different PV parameters on the delay and leakage spreads of a cache. We demonstrated that both the delay and leakage increase as the temperature increases. The cache leakage power and its
STD at 378K are $19.45\times$ and $19.25\times$, respectively, as compared to those at 298K.

We ran simulations for a number of cache and NoC configurations. We demonstrated that a cache with ASG-mode peripheral components and RBGB memory cells has the best balance between delay and leakage power. The delay of the ASG+RBGB implementations is 6.6% larger than that of the SG-mode implementations, while the total leakage power is reduced by 95.2%. For on-chip routers, we showed the importance of performing PVT variation analysis and optimization. If not carefully designed, PVT variations may cause significant uncertainty in router performance, degrading the range of usable operation frequencies. We also presented the power
breakdown of an on-chip router. The dynamic power in the SG+PGFB, IG+PGFB, ASG+PGFB, and ASG+RBGB implementations are 56.4%, 62.8%, 66.0%, and 95.9% of the total power, respectively. We observed that leakage power dominates the power consumption in the input buffer, while dynamic power dominates in the crossbar. Finally, we demonstrated NoC power consumption under synthetic and real traffic loads. PARSEC simulations showed that 77.4% of the power consumption comes from leakage power on an average at 378K.
Chapter 5

Variable-pipeline-stage Router

5.1 Introduction

Next, we present VPSR, a new router design that enables significant runtime power reduction while delivering energy-delay-throughput improvement in the context of FinFET-based interconnection networks. VPSR adjusts the router pipeline latency by varying the back-gate bias voltages of router components and using token-based flit bypassing. The basic idea behind VPSR is to dynamically change the number of pipeline stages from the input ports to the output ports based on incoming traffic. Flits from more accessed ports, which are usually sources of contention, traverse fewer pipeline stages. Flits from less accessed ports, where most router resources are relatively free currently, have to pass through more pipeline stages. Flits possessing tokens may bypass the router pipeline. In other words, different flits may traverse varying-length pipeline stages when passing through the router. This has two advantages. First, router performance is enhanced because VPSR adapts its throughput to network traffic requirement at runtime. Second, significant leakage power can be saved by reverse-biasing the FinFET back gates in infrequently accessed components. This scheme is primarily targeted at FinFETs.
5.2 Critical Path Analysis and Optimization

To optimize both latency and power, we first analyze the critical path from the input port to the crossbar switch in VPSR. Fig. 5.1 shows the critical path traversed by a head flit. It consists of the VC allocator and the LSA, followed by a buffer read (BR) stage, and the GSA. The critical path involves driving data wires in the BR stage and arbiter circuits in the VA and SA stages and, hence, results in significant delay. If the critical path is optimized under normally-biased IG-mode FinFETs, one more pipeline stage has to be added when IG-mode FinFETs in the buffers are reverse-biased. This is accomplished by inserting a flit-wide array of DFFs to separate the GSA stage from the critical path. The increased slack accommodates the delay overhead incurred by reverse-biased IG-mode gates. The crossbar has a dedicated pipeline stage and is designed such that it accommodates the delay overhead incurred by reverse-biased IG-mode gates. For non-critical paths, gate sizes are optimized with reverse-biased IG-mode gates to further reduce their leakage power consumption. In this work, we optimize gate sizes along the critical path with normally-biased IG-mode FinFETs while meeting a target 1GHz clock frequency at the 22nm FinFET technology node.
If the incoming flit is allowed to bypass the router, it is buffered in a flit-wide array of bypass latches. The flit can directly access the crossbar instead of going through the normal pipeline.

### 5.3 Variable Pipeline Stages

Based on the above observations, we propose a router with variable one to four pipeline stages. To illustrate its latency impact, Fig. 5.2 shows the three VPSR pipeline modes through which a flit passes: the normal mode, low power mode, and bypass mode. VA and SA stages execute in parallel in the same cycle. The former two modes allow a router to adjust the number of pipeline stages to incoming traffic’s needs. When a port is heavily accessed, most of the incoming flits can head directly from the BW to the LT stage in three cycles, which corresponds to the normal-mode pipeline. However, in case a port is rarely visited or the power exceeds the budget,
most of the flits have to spend one more cycle in the DFF array before the GSA stage. This leads to four pipeline stages in the router datapath, corresponding to the low power mode. The low power mode results in more slack, which allows the input buffer to be reverse-biased to reduce leakage power. For an input port, VPSR gives a higher priority to the flit stored in the DFF array to access the GSA over the flits in the input buffer, which helps reduce average flit latency and, hence, improves throughput. On the other hand, the GSA gives each input flit an equal chance to access the crossbar. The bypass mode allows flits to bypass the router pipeline by only traversing the ST stage, which improves network latency, energy consumption, and throughput significantly. Lookaheads are sent to the next hop one cycle ahead of data flits in the lookahead link traversal (LA LT) stage. These lookaheads go through the LA control setup stage to perform control setup, reserve router resources, and resolve conflicts to enable bypassing. The bypass mode enables token-based flow control mechanisms and forms the basis for GEVC.

Two modes of SRAM buffer banks are available in our design: normal bank and slow bank. A normal bank corresponds to normally-biased buffers \((V_{HI} = V_{DD} \text{ and } V_{LOW} = 0V)\), whereas a slow bank corresponds to reverse-biased buffers \((V_{HI} > V_{DD} \text{ and } V_{LOW} < 0V)\). In the first scenario, both the input buffer and crossbar are normally-biased, leading to smaller critical path delay and, hence, three pipeline stages. In the second scenario, flits are read from a slow bank and pass through a normally-biased crossbar, leading to four pipeline stages. These two scenarios may coexist under normal conditions, in which some buffer banks are normal while others are slow. In case of a power emergency, when power exceeds an allowed threshold, VPSR switches itself to the power saving mode by reverse-biasing all the buffer banks and crossbar. This minimizes overall leakage power consumption, but increases the number of pipeline stages to four for all flits passing through the router. The crossbar design is based on the matrix-based crossbar [11], which consists of tri-state buffers
at the crosspoints as well as drivers on the input and output paths. The IG-mode logic gates inside the crossbar can also be reversed-biased, which enables the crossbar to be run in the slow mode.

5.3.1 Flow control mechanism

Two flow control mechanisms are adopted in a VPSR-based network: input buffer bank ratio management and ETFC. Input buffer bank ratio management is also the DPM scheme adopted by VPSR, called VPSRDPM, to dynamically reduce the power consumption. VPSRDPM regulates the router performance to just meet the incoming traffic’s need by adjusting the ratio of the number of normal banks to the number of total banks. In contrast to VPSRDPM, ETFC works at the network level to regulate traffic flow. These two flow control mechanisms are explained in detail in the following two sections.

5.4 VPSR Dynamic Power Management (VP-SRDPM)

In this section, we present VPSRDPM, a framework used by VPSR to enable fine-grained DPM. Fine-grained control is made possible by partitioning the input buffers into several banks. Each bank and crossbar can be switched between the normal and slow modes. The switching of bank modes is enabled by ABGB. In order to enable the ABGB technique for VPSRDPM, we incorporate voltage generators into the buffer banks and the crossbar to supply required voltages to the back gates of FinFETs. Details of the voltage generator design are provided in [70]. We assumed a flit size of 128 bits [23]. The size of the voltage generator is $1.225\mu m \times 2.745\mu m$. It is optimized for fast transition (within 1ns or one clock cycle at 1GHz) from normally-biased ($V_{HI} = 0.9V$ and $V_{LOW} = 0V$) voltage levels to reverse-biased voltage levels.
\( V_{HI} = 1.1V \) and \( V_{LOW} = -0.2V \), and vice versa. If the flit size is reduced, the sizes of the router and voltage generator can be scaled down as well. It is possible to replace the voltage generator with a simple power switch that selects one of only two voltage levels. However, when process and temperature variations occur and lead to excessive leakage, higher levels of reverse-bias voltages may be necessary, such as \((V_{HI} = 1.3V\) and \( V_{LOW} = -0.4V\)). This favors the use of a voltage generator.

ABGB provides the crucial foundation for performing DPM in FinFET-based circuits because it is very effective and easy to implement. While popular techniques like dynamic voltage scaling (DVS) \([165]\) can reduce active-mode power, they incur significant transition latency and energy overhead. ABGB, on the other hand, can switch the back-gate bias voltages of FinFETs very quickly, with little transition energy overhead. Since energy is consumed every time the output of the voltage generator is switched, a corresponding amount of leakage energy should be saved in FinFETs to reach the breakeven point. For VPSR, the breakeven point is only four clock cycles. VPSRDPM updates the back-gate bias voltages only every 1,000 cycles. Thus, the energy saved by VPSR far outweighs the energy consumed by the voltage generator. The power consumption of the voltage generators is included in that of the router components in our simulations.

VPSRDPM periodically updates the ratio of the number of normal banks to the number of total banks (normal-to-total bank ratio) based on the incoming traffic condition to proactively regulate the power consumption while meeting the traffic requirement. After the voltage transition, the voltage generator is turned off by disconnecting it from the power supply. Therefore, power is only dissipated during the voltage transition period. The bias voltage is maintained on the capacitance at the output of the voltage generator for a while. The voltage generator needs to be turned on periodically in order to recharge the capacitance. Any online power estimation method can be used, such as the one in \([36]\). When the average power
Fig. 5.3: Normal/slow bank structure

exceeds the power budget, VPSRDPM switches the router to the power emergency mode.

5.4.1 Normal bank and slow bank structure

Fig. 5.3 shows the microarchitectural block diagram of the input buffer and buffer controller in Fig. 5.1. As mentioned earlier, the input buffers are partitioned into normal banks and slow banks for fine-grained DPM. The motivation behind this is that memory cells are usually significant consumers of leakage power. In conventional memory cell arrays, each read or write operation only involves one row of memory cells, while the other rows are idle and dissipate leakage power. Thus, we partition the entire pool of memory into several banks with different bias voltages. The normal banks are biased at $V_{HI} = 0.9V$ and $V_{LOW} = 0V$, while the slow banks are biased at $V_{HI} = 1.1V$ and $V_{LOW} = -0.2V$. Although the banked-buffer scheme requires some
extra wires and bitline decoders, the leakage power saved from the slow banks still outweighs the power consumed by them.

Multiplexers and demultiplexers, which are controlled by the buffer controller, are used for controlling bank write and read operations. As shown in Fig. 5.3, each bank is shared among all VCs. The VC flit buffers are dynamically allocated across the banks. A buffer allocation table (BAT) records the locations of the flits of each VC in different banks. At low load, when most of the memory banks are free, VPSR sets most of its banks to slow banks. To avoid throughput degradation, normal banks are given higher priority to slow banks. Thus, if both normal banks and slow banks are available for an incoming flit, it will be written into a normal bank. When a flit is read from the input buffer, the buffer controller notifies VPSR as to which type of bank the next flit is coming from. If the flit is read from a normal bank, it is forwarded directly to the crossbar switch. Otherwise, it is buffered in the DFF array first and then forwarded to the crossbar in the following cycle. Thus, VPSR is able to change the number of its pipeline stages based on this information. To utilize the slack time during which the selected flit is read from the slow buffer to the DFF array, the buffer controller may choose another flit from a normal bank heading to the same output port to access the crossbar. A dual-port SRAM structure is used as a conventional input buffer to enable simultaneous write and read accesses of the bank. Thus, when a flit is read, the incoming flit is allowed to write into another memory address in parallel.

5.4.2 Input buffer bank ratio management

The aim of input buffer bank ratio management in VPSRDPM is to determine the best normal-to-total bank ratio. A good bank ratio would efficiently use bank resources to achieve high performance, but leave unused banks in the slow mode. The traffic patterns in CMPs do not always result in a high utilization at all input ports. This
Figure 5.4: VPSR flow control

means that for some accessed ports, just a fraction of the banks needs to be in the normal mode to accommodate incoming traffic. The other banks can be switched to slow mode to save leakage power. VPSRDPM dynamically adjusts the normal-to-total bank ratio every 1,000 cycles to adapt to the traffic requirements at runtime. Meanwhile, the voltage generators of the unaffected banks are also turned on to recharge the capacitances that sustain the bias voltages. Thus, most flits spend the smallest possible time in the router pipeline. This allows the router’s throughput to approach the all-normal-bank case, yet save a significant amount of leakage power.

Fig. 5.4 shows an example of how the input buffer bank ratio management works in VPSRDPM. Each unit on the horizontal axis is the update period (1,000 cycles). In the beginning, there is only one normal bank in each input buffer, the others being slow banks. Every 1,000 cycles, VPSRDPM estimates the buffer utilization and power consumption over the previous 1,000 cycles, and adjusts the normal-to-total bank ratio. The four scenarios that VPSRDPM has to handle at runtime are discussed next.

**Bandwidth required:** This scenario occurs when the incoming traffic is faster than
the outgoing traffic, which results in flits getting stuck in the input buffer. The existing normal-to-total bank ratio is too small to handle the traffic load. Hence, one slow bank is changed to a normal bank to enable the increased throughput to clear out the flits in the slow bank.

**Bandwidth satisfied:** This scenario occurs when the number of outgoing flits is equal to or more than the number of incoming flits. Thus, the normal-to-total bank ratio is more than required. To avoid excessive leakage power, the flow control mechanism decrements the number of normal banks by one, thus reducing throughput. However, to maintain the performance under light traffic, VPSR keeps at least one normal bank at each input buffer, if there is no power emergency.

**Power emergency:** The power emergency scenario occurs when the total router power exceeds the power budget. This budget is usually set by system cooling requirements, and is regulated by the operating system. The purpose of this budget is to prevent the system power from exceeding the limit that the system can support. VPSRDPM responds to this scenario by reverse-biasing the back gates of the FinFETs in all the memory banks as well as the crossbar switch. Hence, the leakage power and dynamic power are both greatly reduced.

**Fast restart:** In order to restore the router throughput quickly after a power emergency, VPSR has to readjust the normal-to-total bank ratio to meet the incoming traffic pattern. To do this, VPSRDPM remembers the normal-to-total bank ratio used right before the power emergency period. Each time, when restoring from the power emergency mode, VPSR sets its normal-to-total bank ratio to half the ratio right before the power emergency. This jump-to-half scheme has two advantages. First, VPSR does not have to restart with all slow banks, thus its throughput can be restored quickly. Second, if the incoming traffic pattern changes, VPSR can re-adapt to the new traffic pattern instead of directly jumping to the all-normal-bank mode.
Fig. 5.5 shows the flowchart of the VPSRDPM flow control mechanism. We have two counters for each input buffer: one counts the incoming flits and the other counts the flits read from the input buffer. Online router power estimation [3, 6] is used to estimate the power consumption in the previous cycle. Every update period, which is equal to 1,000 cycles, the number of buffer writes is compared with the buffer reads at each input port. If there are more flits written into the buffer than read out, the number of normal banks is incremented, otherwise it is decremented. In parallel, the average power is calculated for the past ten update periods to avoid an instantaneous power overshoot. If the average power is greater than the power budget, VPSR enters the power emergency mode, and performs a fast restart after another update period.

5.5 Enhanced Token Flow Control Mechanism

In this section, we present the concept of ETFC, which is the network-level traffic flow control scheme used in VPSR-based networks. ETFC is enhanced from token flow control (TFC) [23] to provide better buffer utilization and enhanced usage of guaranteed tokens. We also explain in detail the conditions under which different types of
tokens are used, as well as their priorities and communication protocols. We propose a new concept of using guaranteed tokens to establish temporary EVCs to quickly bypass packets from congested regions. Lastly, we show the router microarchitecture and analyze the overheads.

5.5.1 Token forwarding in networks

Fig. 5.6 demonstrates the token forwarding mechanism in ETFC. Similar to TFC, both normal tokens and guaranteed tokens are incorporated. Normal tokens are sent to the neighboring nodes of the source router and broadcast to signal the availability of free VCs or flit buffers, while guaranteed tokens are forwarded to congested regions by unicast. ETFC differs from TFC in two aspects. First, normal token forwarding takes place for only up to two hops. Any flit that intends to bypass more than two hops needs to acquire a new token to allow chaining of multiple token routes. Second, a guaranteed token provides a guarantee to a single packet for resource availability at the endpoint node of the token route, rather than enabling the bypassing of a single flit as in TFC. This scheme is similar to having a temporary EVC for the entire packet, and is called GEVC. A GEVC guarantees that all the flits belonging to a packet can bypass the token route up to three hops, and is automatically turned off after the tail flit of the packet traverses the route. In Fig. 5.6 a normal token is sent from the west port of node 11 to node 10, and is then broadcast to nodes 6, 9, and 14 (assuming that the north, west, and south ports of node 10 also have a sufficient number of free VCs and buffers to send out tokens). Thus, the SE token at node 6 can be used by a flit to travel through the south output to node 10, followed by the east output to reach node 11. For the congested region in Fig. 5.6 there is no such normal token available due to lack of router resources. To relieve network congestion, a guaranteed token is sent from the north port of the non-congested node 12, and is unicast through nodes 8 and 4 to destination node 3. These nodes together form a
GEVC. A packet in node 3 can acquire the guaranteed token, skip through congested nodes 4 and 8 via the GEVC, and directly go to node 12 (assuming that node 12 is closer to the packet’s destination).

Normal tokens are not forwarded more than two hops in ETFC. Bypassing flits through more hops requires chaining of multiple token routes or using a GEVC. The reason is threefold. First, the source router only reserves resources for a round-trip delay between adjacent nodes, which typically include four free VCs and buffers. For nodes farther than one hop, a normal token only acts as a hint of resource availability at its source node. Taking the normal token scenario in Fig. 5.6, for example, if nodes 6, 9, and 10 all send data flits to node 11 using normal tokens, the free VCs and buffers at the west port of node 11 will deplete quickly, forcing node 11 to turn off the normal token at its west port. Therefore, the farther a node is from the source node of the token, the less likely its flits can find a bypass path all the way to the
end of the token route. Second, the longer the token route, the more cycles it takes to turn off the token. The off signal takes $d_{\text{max}}$ cycles to travel $d_{\text{max}}$ hops to the farthest node, which takes one more cycle to process the off signal. Thus, a total of $d_{\text{max}} + 1$ cycles are required to turn off a normal token. Third, longer token routes imply larger token and lookahead sizes, leading to extra wiring overhead. According to [23], if $d_{\text{max}} = 2$, the per-port wiring overhead for tokens ($W_{\text{normal}}$) and lookahead signals ($W_{\text{lookahead}}$) are 4 and 3 bits, respectively. If $d_{\text{max}} = 3$, $W_{\text{normal}}$ and $W_{\text{lookahead}}$ become 10 bits and 6 bits, respectively. The wiring overhead grows exponentially as $d_{\text{max}}$ increases.

ETFC provides two types of normal tokens: Vtokens for VCs and Btokens for buffers. A Vtoken provides a hint for both VC and buffer availability at an input port of a router, while a Btoken only implies buffer availability. In TFC, all tokens are Vtokens. A normal token is sent only if there are greater number than a fixed threshold $b_{\text{thr}}$ of buffers and $v_{\text{thr}}$ of VCs available at a particular input port. However, this design may result in poor utilization of buffer resources. Assume that all of the available VCs at an input port are occupied, but each VC only contains a single flit. In such a scenario, no normal tokens can be sent even though the free buffers are still abundant at that input port. In order to efficiently utilize buffer resources, ETFC also provides Btokens to signal buffer availability. A Btoken is sent if an input port has a greater number than $b_{\text{thr}}$ free buffers, even if there are no free VCs available. Therefore, in ETFC, a head flit may only acquire a Vtoken, but a body or a tail flit may use either a Vtoken or Btoken. Note that if a node is operating under a power emergency, no token is turned on from that node. However, it is still able to forward guaranteed tokens arriving from its neighbors.

It is important to take the ordering of flits into account when the bypassing flits belong to the same packet. However, in a network which allows flit bypassing, it is possible that a tail flit bypasses a node in which the body flits of the same packet are
still resident. Fig. 5.7 shows two scenarios in which a tail flit bypasses a router. In the first scenario, the VC is empty. The tail flit bypasses the router and releases the VC without causing any problem. In the second case, the VC contains the body flit that precedes the tail flit and, hence, allowing direct bypassing of the tail flit breaks the flit order of the packet and should be avoided. TFC does not handle this scenario.

To maintain the flit order while keeping the benefits of flit bypassing, ETFC adopts a push-and-pop concept in the second scenario. When a router receives the lookahead of a tail flit and finds that there are body flits of the same packet in its VC, the router still grants flit bypass to that VC for the next cycle. However, in the next cycle, the foremost body flit in the VC is popped and sent to the crossbar switch, while the incoming tail flit is pushed into the VC and stored. The popped body flit takes over the token from the tail flit and continues flit bypassing until it arrives at the end of the token route. Note that in the second scenario, if the body flit is stored in a slow bank, it is first read into the DFF array at that input port during the LA control setup stage. As soon as the tail flit arrives, the body flit traverses the crossbar switch without any extra delay.

### 5.5.2 GEVC protocols

In ETFC, a guaranteed token is used to establish a GEVC, providing a guarantee to a single packet rather than unrelated flits along the token route. Fig. 5.8 illustrates
Figure 5.8: Original guaranteed token problem

a scenario explaining why ETFC employs this restriction. Suppose node 4 sends out a guaranteed token through congested nodes 3 and 2 to node 1. If all three nodes 1, 2, and 3 use the guaranteed token and send flits to node 4, the free buffers at the west port of node 4 will be depleted quickly. In such a scenario, only the head flit of node 1 reaches node 4. The other body flits are left behind at node 1, occupying router resources and blocking other incoming flits. If node 1 does not release the VC, the bottleneck is not removed and the area remains congested. Therefore, a better approach is to clear the congested flits first at node 1 by allowing the entire packet to bypass the congested network to node 4 via a GEVC. As soon as the packet leaves node 1, the VC is released, allowing another packet from the upstream node to access it. Since a guaranteed token can only be used by a single packet, the source router of the guaranteed token only needs to ensure a free VC and enough buffers for a packet. In contrast to ETFC, TFC has to reserve nine free VCs and buffers at an input port to send out a guaranteed token if $g_{\text{max}}$ equals three [23], which is not that efficient. A GEVC can only be used by nodes that are two or three hops away from the source node. Nodes that are adjacent to the source router may choose either a regular pipeline or normal tokens. Another advantage of GEVCs is that there is no need to spend extra cycles to do off signaling. The GEVC turns itself off automatically as the tail flit of the packet leaves the intermediate routers.
The main purpose of GEVCs is to detour packets from congested regions or nodes that are operating under a power emergency. Therefore, a GEVC is used only under restricted conditions, acting as the last stand to control traffic flow. If a packet is not able to obtain any normal tokens to make progress and satisfies these conditions, it may look for an available guaranteed token to establish a GEVC to get it closer to its destination. There are two conditions that a packet needs to satisfy to acquire a guaranteed token. First, the VC flit buffer of the packet at the input port has to be at least half-filled. This restricts the usage of GEVCs only to packets that are ready to transmit. Second, the buffer utilization of the corresponding input port has to exceed a threshold of $b_{GEVC}$. This ensures that only heavily-accessed input ports are entitled to use guaranteed tokens.

Fig. 5.9 shows an example of an input port of a node possessing a guaranteed token. All three VCs, VC 1, VC 2, and VC 3, contain packets that intend to acquire the guaranteed token to establish a GEVC. There are totally nine flit buffers in the input buffer, and five of them are occupied, meeting the second condition for using the guaranteed token. Among the three VCs, only VC 3 is fully filled. Each of the other two VCs only has one flit in the VC flit buffer. As a result, if a normal token is not available, VC 3 may choose to use the guaranteed token to allow its packet to proceed via the GEVC.
Similar to TFC, there are VC and buffer thresholds for turning on a guaranteed token at an input port in ETFC. As mentioned above, an input port needs to ensure the availability of one free VC and enough buffers to accommodate a packet. Meanwhile, the input port itself has to maintain its normal token operations as well. Given that an input port needs to reserve a minimum amount of $v_{thr}$ VCs and $b_{thr}$ buffers for normal token operations, the VC and buffer thresholds for turning on a guaranteed token, $v_{gthr}$ and $b_{gthr}$, are given by the following equations:

$$v_{gthr} = v_{thr} + 1$$  \hfill (5.1)

$$b_{gthr} = b_{thr} + b_{packet}$$  \hfill (5.2)

where $b_{packet}$ is the number of buffers required to accommodate a packet. Once the guaranteed token is turned on, the input port keeps the VC and $b_{packet}$ buffers reserved for a fixed number of cycles, which equals the $g_{max}$-hop round-trip delay. If no lookahead carrying the guaranteed token arrives within this period, the guaranteed token expires automatically and the reserved router resources are released. The input port then assumes that there is no traffic congestion or power emergency at all in that direction, and stops sending any guaranteed tokens for a period of time $t_{sleep}$. After this period, the input port may turn on the guaranteed token again if it still has enough resources.

Flits bypassing a node via a GEVC have the highest priority to access the crossbar switch. A lookahead carrying a normal token that conflicts with a lookahead using a GEVC for the same router output port is killed. Conflicts between GEVCs, e.g., lookaheads arriving at a router simultaneously from different input directions via GEVCs and heading for the same output port, should be proactively avoided by restricting the forwarding direction of guaranteed tokens. Guaranteed tokens are forwarded to the highest utilized port that is not currently part of a GEVC. Fig. 5.10
Figure 5.10: Guaranteed token forwarding example

shows an example. There is a GEVC from the east port to the south port. If a guaranteed token arrives at the west port, it cannot be forwarded to the east port because the east port is already reserved for an existing GEVC. It can only be forwarded to either the north or the south port, which is available for establishing a GEVC. The input buffer of the north port is more utilized than that of the south port, indicating that the north direction is probably more congested. Therefore, the guaranteed token is forwarded to the north port. ETFC adopts a similar starvation avoidance mechanism as TFC. Each node keeps track of the number of consecutive flits bypassed for each output port. If this number for an output port exceeds a threshold $str_{off}$, that output port enters a starvation-avoidance mode. All incoming lookaheads heading to that output port with normal tokens are killed, except for the lookaheads using GEVCs. To further prevent starvation caused by GEVCs, a node does not forward any guaranteed tokens coming from an output port in the starvation-avoidance mode, thereby preventing new GEVCs from being established. The starvation-avoidance mode lasts for $str_{cycle}$, after which the output port returns to normal operation.
GEVC provides a dedicated bypassing path for a packet. However, too many GEVCs may deteriorate the throughput and latency of the normal flits because they are required to give way to flits on the bypassing path. Thus, for an effective trade-off, selective use of GEVCs is necessary. In the 16-node case depicted in Fig. 5.11, we require that only the corner and some of the edge routers be eligible to issue guaranteed tokens in light of the fact that central regions are usually heavily congested. More sophisticated algorithms that require routers to rotate the issuance of guaranteed tokens are possible. However, this is left as future work.

5.5.3 Router microarchitecture

Fig. 5.11 shows the router microarchitecture, which incorporates several modifications to the baseline design. The components that have been modified include the input buffers, RC unit, SA, token table, and GEVC controller. The input buffer structure
was shown in Fig. 5.3. It consists of normal and slow banks as well as a buffer controller, as explained earlier. The bypass latch temporarily holds the flit to be bypassed. The GEVC buffers are dynamically allocated and only exist at the end of the GEVC route (i.e., the source router of the guaranteed token). The RC unit determines if a normal token or a guaranteed token is granted to an incoming flit while calculating its forward route. A switch port priority vector is added to the switch allocator. A separate token table is included to record the available normal and guaranteed tokens to be used in the router. The GEVC controller checks if any input port satisfies the conditions to turn on a guaranteed token, and determines to which port a received guaranteed token is forwarded. The GEVC controller also monitors the round-trip time to determine when the reserved resources are to be released due to the expiration of guaranteed tokens. Note that these modifications mainly affect the control logic and arbiters, which account for a very small part of router complexity [23].

ETFC incurs additional wiring overhead due to the use of lookaheads and tokens between nodes. The per-port wiring required for normal tokens, guaranteed tokens, and lookaheads is discussed next.

**Normal tokens:** In ETFC, $d_{\text{max}}$ is restricted to two hops. For each hop, two bits are used to encode the four input port directions in each router. An additional bit labels the token to be either a Vtoken or Btoken. As a result, to send a normal token two hops away from the source router, six bits are required. An extra wire is included at each port to indicate the on/off state of the token. Therefore, the per-port wiring overhead for normal tokens, $W_{\text{normal}}$, is seven bits.

**Guaranteed tokens:** In contrast to normal tokens, each node records the direction in which the received guaranteed tokens are forwarded. Only one additional wire is necessary for each port to indicate if the port is reserved for a GEVC. Therefore, the per-port wiring overhead for guaranteed tokens, $W_{\text{guaranteed}}$, is one bit.
**Lookaheads:** A lookahead in ETFC needs to encode the flit’s route for an additional $d_{\text{max}} - 1$ hops. Using two bits to encode the four router output ports, three bits for VC IDs and one bit for valid bit, the per-port wiring overhead for lookaheads, $W_{\text{lookahead}}$, is given by:

$$W_{\text{lookahead}} = 6 \cdot (d_{\text{max}} - 1) = 6$$  \hspace{1cm} (5.3)

From the above discussions, the wiring overhead in ETFC is only 14 bits per port in all, which is small compared to the data channel width.

The area and power overheads of the router are small relative of those of the baseline design. The voltage generators occupy 0.01% of the total area of the router. The extra flit-wide arrays of DFFs constitute 0.62% of the total router area. The online power estimation circuits only involve counters for each router port and inputs of the crossbar [36]. For a router with five ports, ten counters are required. Each counter consists of twenty DFFs. Therefore, the power estimation circuits contain two hundred DFFs in all. These occupy 0.20% of the total router area. If SG-mode FinFETs and PGFB memory cells are used, the extra latches and power control circuitry consumes 2.04% of the input buffer power, which corresponds to 0.65% of the router power.

### 5.6 Experimental Results

In this section, we present experimental results for the proposed VPSR design. The network latency under different injection rates is explored first. Then, the comparison of network power consumption for different traffic patterns is presented. The power breakdown of the network components is explored for four different cases. Finally, PARSEC simulation results are presented.
Table 5.1: Network parameters

<table>
<thead>
<tr>
<th>Network parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>1 GHz</td>
</tr>
<tr>
<td>Technology</td>
<td>22nm FinFET</td>
</tr>
<tr>
<td>Topology</td>
<td>4×4 Mesh</td>
</tr>
<tr>
<td>Link length</td>
<td>1 mm</td>
</tr>
<tr>
<td>Router ports</td>
<td>3~5</td>
</tr>
<tr>
<td>Flit size</td>
<td>128 bits</td>
</tr>
<tr>
<td>Number of message classes</td>
<td>4 classes</td>
</tr>
<tr>
<td>VCs per message class</td>
<td>8 VCs</td>
</tr>
<tr>
<td>Buffers per port</td>
<td>48 (dynamically allocated)</td>
</tr>
<tr>
<td>Link latency</td>
<td>1 cycle</td>
</tr>
<tr>
<td>Token $d_{\text{max}}$</td>
<td>2 hops</td>
</tr>
<tr>
<td>Token $g_{\text{max}}$</td>
<td>3 hops</td>
</tr>
<tr>
<td>Token $\text{str}_{\text{off}}$</td>
<td>20 cycles</td>
</tr>
<tr>
<td>Token $\text{str}_{\text{cycle}}$</td>
<td>3 cycles</td>
</tr>
<tr>
<td>$v_{\text{thr}}$ &amp; $b_{\text{thr}}$</td>
<td>4 buffers</td>
</tr>
<tr>
<td>$v_{\text{gthr}}$</td>
<td>5 buffers</td>
</tr>
<tr>
<td>$b_{\text{gthr}}$</td>
<td>8 buffers</td>
</tr>
<tr>
<td>$t_{\text{sleep}}$</td>
<td>10 cycles</td>
</tr>
<tr>
<td>Routing algorithm</td>
<td>Dimension-ordered X-Y</td>
</tr>
</tbody>
</table>

5.6.1 Simulation setup

GARNET-FinFET [70] is used to model the power and performance of VPSR. GARNET-FinFET supports the VPSR structure, allowing the use of normal and slow banks, and enables its flow control mechanism.

Table 5.1 shows the network parameters along with the routing algorithm used. These parameters are chosen such that the network and router configurations are the same as those used in [23] to enable comparison between VPSR and TFC. These numbers are fed to GARNET-FinFET. If not specifically stated, the operating temperature is assumed to be 105°C. The number of router ports ranges from three to five because the routers in the corner and along the sides have fewer input ports than the routers in the middle of the mesh. Both synthetic and real traffic patterns are used in our simulations. For synthetic traffic simulations, the number of simulation cycles is
100,000 and includes 1,000 cycles for warm-up. The synthetic traffic patterns contain a mix of single-flit control packets and four-flit long data packets spread across four protocol message classes. PARSEC [103] is used to obtain real traffic patterns.

5.6.2 Network latency vs. injection rate

Fig. 5.12 plots network latencies for VPSR, VPSR with power budget, TFC, and ETFC alone as a function of packet injection rate, comparing them with the baseline case. The power budget of each router is set to the average power of the VPSR case; thus, the power consumed in VPSR is further reduced. The traffic pattern used in this case is uniform random, in which packets are sent to destinations in a uniform random fashion. The ETFC-alone case provides the smallest network latency. The network latency of the VPSR case is slightly higher than that of the TFC case because of the low-power-mode pipeline. The VPSR-with-power-budget case has slightly longer network latency than VPSR due to the use of the power emergency mode. When the
injection rate is 0.1 packets/node/cycle, VPSR and VPSR with power budget reduce network latency by 46.1% and 41.8%, respectively, relative to the baseline.

5.6.3 Power comparison for different traffic patterns

Fig. 5.13(a) compares the power consumption of the entire network for the baseline, VPSR with power budget, TFC, VPSR, and ETFC alone for three types of traffic patterns. Fig. 5.13(b) shows the power reduction obtained. These traffic patterns include uniform random, tornado, and bit-complement traffic [126], and were generated by GARNET-FinFET. Different traffic patterns have different link and router port utilization rates, thus dissipate different amounts of power. The packet injection rate is fixed to 0.1 packets/node/cycle. On an average, VPSR is able to save 21.5% of power relative to the baseline. Moreover, with a power budget, VPSR can reduce power by 27.5% for the tornado traffic, and 26.3% on an average. TFC reduces power by 13.7%, on an average, which is lower than the reported value in [23]. The reason is that power consumption for technologies beyond the 22nm node is dominated by leakage power. Although TFC is able to reduce the dynamic power significantly by flit bypassing, it does not reduce leakage power. The ETFC-alone case can only reduce power by 15.2%, on an average, due to the same reason. VPSR achieves a higher power reduction because it reduces leakage power as well. From the above experiments, it can be seen that VPSR provides not only low network latency, but also significant power reduction.

5.6.4 Comparisons under different routing schemes

Table 5.2 compares the network latency (cycles) for two routing schemes. We compare dimension-ordered routing (DOR) with adaptive minimal west-first routing (WFR). In DOR, a packet is routed in one dimension until it needs to make a turn. As a result, in a mesh network, DOR only allows one turn. In WFR, a packet is routed west first,
Figure 5.13: Power consumption for various types of network traffic
Table 5.2: Network latency (cycles) under two routing schemes

| Scheme       | Uniform random |  |  |  |  |  |  |
|--------------|----------------|----------------|
|              | DOR | WFR | DOR | WFR | DOR | WFR |
| Baseline     | 35.27 | 25.35 | 33.12 | 24.82 | 29.42 | 130.36 |
| TFC          | 32.38 | 32.38 | 30.99 | 28.22 | 25.86 | 178.70 |
| VPSR         | 32.25 | 31.87 | 31.39 | 28.52 | 26.62 | 137.42 |
| ETFC alone   | 31.94 | 31.51 | 30.74 | 27.99 | 25.86 | 152.48 |

if necessary, and then adaptively south, east, and north. The packet injection rate is 0.04 packets/node/cycle. We can see that the performance of adaptive minimal WFR is better than that of DOR for uniform random and tornado traffic. However, because of saturation in the bit-complement case, its network latency is worse than that of DOR. When using WFR, the latencies of the baseline router in the uniform random and tornado cases are reduced by 28.1% and 25.1%, respectively. TFC, VPSR, and ETFC-alone cases perform better than the baseline case when the routing scheme is DOR, however, result in longer latencies when the routing scheme is WFR. This is because WFR is adaptive, in which the flit routing paths are not as regular as in the case of DOR. Therefore, bypassing flits in such a routing scheme may block the traversal of regular flits. Although WFR provides better performance, it saturates faster than DOR. We can also see that ETFC outperforms or equals TFC in all cases.

5.6.5 Comparisons under different input buffer sizes

Table 5.3 compares the network latencies for various per-port input buffer sizes. The packet injection rate is 0.04 packets/node/cycle. When the input buffer size is reduced, the network latency increases due to insufficient capacity to store the flits at each input port. The VPSR latency increases by 33.1% when the input buffer size decreases from 48 flits to 12 flits. This is due to reduced number of banks. In the VPSR case, each bank contains 12 flits. Therefore, when the input buffer contains
Table 5.3: Network latency (cycles) under different input buffer sizes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>48 flits</th>
<th>24 flits</th>
<th>12 flits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>35.27</td>
<td>35.29</td>
<td>38.51</td>
</tr>
<tr>
<td>TFC</td>
<td>32.38</td>
<td>32.41</td>
<td>35.24</td>
</tr>
<tr>
<td>VPSR</td>
<td>32.25</td>
<td>32.26</td>
<td>42.92</td>
</tr>
<tr>
<td>ETFC alone</td>
<td>31.94</td>
<td>31.95</td>
<td>34.83</td>
</tr>
</tbody>
</table>

Table 5.4: Network latency (cycles) under different numbers of VCs

<table>
<thead>
<tr>
<th>Scheme</th>
<th>8</th>
<th>5</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>35.27</td>
<td>35.28</td>
<td>36.67</td>
</tr>
<tr>
<td>TFC</td>
<td>32.38</td>
<td>32.38</td>
<td>35.11</td>
</tr>
<tr>
<td>VPSR</td>
<td>32.25</td>
<td>32.25</td>
<td>35.75</td>
</tr>
<tr>
<td>ETFC alone</td>
<td>31.94</td>
<td>31.94</td>
<td>34.90</td>
</tr>
</tbody>
</table>

only 12 flits (i.e, there is only one bank), it can only switch between normal mode and slow mode. To the contrary, when the input buffer contains four banks, it enables VPSR to regulate the normal-to-total bank ratio such that it meets the traffic load requirements.

5.6.6 Comparisons under different numbers of VCs

Table 5.4 compares the network latencies under various numbers of VCs per message class at each input port. The number of VCs per message class is varied while keeping the number of message classes the same. Sharing of VCs between message classes is enabled. We can see that as the number of VCs is reduced, the network latency increases or stays the same for all cases. This is because reducing the number of VCs may increase the wait time for flits to be allocated a free VC.
5.6.7 Router power breakdown

Fig. 5.14 presents the power breakdown of the routers under the tornado traffic for the five cases. The power breakdown includes contributions from buffers, crossbar, arbiters, and local clock tree. VPSR reduces buffer power by 47.7%. Under a power budget, VPSR further reduces buffer and crossbar power by 9.6% and 6.1%, respectively, leading to a total power reduction of 27.5%. Note that the ETFC-alone and TFC cases reduce buffer power by 38.9% and 33.2%, respectively, because there is no buffer write and read during flit bypassing. The crossbar power is lower for VPSR with power budget, but its reduction is not as significant as that of the buffer. This is because the crossbar consists of long wires that contribute huge wire loads, such that its power is dominated by dynamic power. Reverse-biasing the FinFETs does not reduce dynamic power. On the other hand, buffer power is mainly dominated by leakage. Therefore, it is significantly reduced by back-gate biasing.
5.6.8 Real traffic simulation

Fig. 5.15 presents PARSEC simulation results for network power for the baseline, TFC, and VPSR. We use a cycle-accurate CMP simulator GEMS \cite{122} to model the 16-core CMP. Cores have private 64KB L1 instruction and 64KB L1 data caches and share a distributed 4MB L2 cache. The caches are kept coherent using a MOESI directory-based protocol. Each core is connected to a $4 \times 4$ mesh with DOR.

Ten PARSEC benchmarks are included in the simulation. Only critical regions of the benchmarks are simulated. Each of the PARSEC benchmarks contains three regions: initiation, critical, and ending. The initialization region deals with initialization of the program parameters and declaration of the program modules, which are not relevant to network activities. On the other hand, the ending region deals with memory space releasing and generation of the output files. If these two regions are included, the network statistics (e.g., packets/total cycles) are underestimated.
Measuring the critical regions only ensures that the network statistics are practically estimated.

The network latency is not shown since PARSEC benchmarks do not inject traffic that is as heavy as in the synthetic cases. The differences in network latency are thus caused by the differences in pipeline depth, which is independent of the trace. We can see that both TFC and VPSR consume less power than the baseline case. On an average, TFC and VPSR reduce power consumption by 8.2% and 19.7%, respectively, relative to the baseline case. The power emergency mode is not activated due to the low traffic injected by PARSEC benchmarks.

5.7 Chapter Summary

We presented an in-depth design of a new router architecture, VPSR, suited to FinFET-based interconnection networks. VPSR targets power reduction while maintaining high network throughput. We proposed VPSRDPM to regulate the normal-to-total bank ratio for input buffer banks. We used voltage generators to enable fine-grained DPM through ABGB. We presented a detailed implementation of ETFC, which allows packets to use normal tokens and GEVCs to find routes along which intermediate nodes can be bypassed. We used GARNET-FinFET, which provides a complete platform for computer architects to quickly estimate the power of FinFET-based interconnection networks at an early design stage. Experimental results show that VPSR is able to significantly reduce power consumption by adapting to traffic, with little latency overhead.
Chapter 6

Conclusions and Future Work

In this chapter, we present the dissertation conclusion and possible future directions.

6.1 Conclusions

We presented a FinFET design library to model delay, power, and area of FinFET logic gates and memory cells, under PVT variations, considering the effect of spatial correlations. The FinFET design library is based on detailed device simulations using mixed-mode TCAD. It consists of FinFET logic and memory cell libraries. The FinFET logic library contains various logic gates with different sizes and FinFET styles. The FinFET memory cell library contains four types of memory cells: PGFB, RBGB, 4T SRAM, and 8T SRAM. The FinFET design library is modular, and can be extended to incorporate new designs of logic gates and memory cells. At the circuit level, it specifies the capacitance and leakage current values of logic gates and memory cells for various FinFET operation modes and at different temperatures. We developed a circuit-level delay model for analyzing gate delay. The delay model calculates the delay spread for any given circuit. For leakage power, we derived a macromodel for each type of logic gate and memory cell. The macromodels are derived based on empirical parameters extracted from TCAD device simulations.
In both models, we used the rectangular grid-based method to model the spatial correlation of the PV parameters. It hierarchically partitions a die into multiple levels of grids. This method can be easily adapted to caches and NoCs, which are inherently organized in a hierarchical grid structure. We validated the FinFET design library with a number of experimental results.

We presented FinCANON, which is an integrated framework for modeling delay, power, and PVT variations for FinFET-based caches and NoCs. The FinCANON framework consists of two major parts: CACTI-PVT for caches, and ORION-PVT for NoCs. Both parts are capable of modeling PVT variations based on the specified extent of variations. The cache and NoC components are represented with macro-models. We extended GARNET to GARNET-FinFET, which is a cycle-accurate power/performance simulator for FinFET-based NoCs. We ran Monte-Carlo simulations for a number of cache and NoC configurations with different capacities, temperatures, logic gate styles, supply voltages, process variation parameters, and memory cell styles. From the simulation results, we observed the impact of different PVT parameters on delay and leakage power. We also demonstrated that a cache with ASG-mode peripheral components and RBGB memory cells has the best balance between delay and leakage power. Moreover, we showed that larger cache capacities or larger memory cells lead to more delay and leakage power variations. At the NoC level, we ran ORION-PVT to estimate the delay and power of different router components. In addition to router components, the power consumption of the clock tree, links, and DFF arrays were also estimated. We applied FinCANON to NoCs and experimented with PARSEC real-traffic benchmarks. The power consumption of the entire NoC can be obtained by summing the power of input buffers, crossbar, arbiters, clock tree, and links. Experimental results demonstrated that leakage power can be significantly reduced by using ASG-mode FinFETs. FinCANON enables computer architects to quickly evaluate the impact of PVT variations on FinFET-based
caches and NoCs at an early design stage, and quantify tradeoffs among different architectures.

We presented a VPSR architecture for FinFET-based NoCs to enable significant runtime power reduction while delivering energy-delay-throughput improvement. VPSR dynamically adjusts its performance to meet the current traffic requirement. In VPSR, flits from more accessed ports, which are usually sources of contention, traverse fewer pipeline stages. This has two advantages. First, router performance is maintained because VPSR adapts its throughput to network traffic at runtime. Second, significant leakage power can be saved by reverse-biasing the FinFET back gates in infrequently-accessed components. This scheme is only possible for FinFETs.

We investigated a fine-grain DPM technique, called ABGB, to exploit the leakage power reduction made possible by IG-mode circuits. ABGB dynamically controls the back-gate bias voltages of FinFETs. In order to exploit ABGB for VPSR, we incorporated voltage generators into the buffer banks and crossbar. While popular techniques like DVS can reduce active-mode power, they incur significant transition latency and energy overheads. ABGB, on the other hand, can switch the back-gate bias voltages of FinFETs very quickly, with little transition energy overhead. Thus, it can be applied in a very fine-grain manner. For VPSR, the breakeven point is only four clock cycles. Its flow control mechanism updates the back-gate bias voltages only every 1,000 cycles. Thus, the energy saved by VPSR far outweighs the energy consumed by the voltage generators.

We also presented ETFC, a flow control mechanism that improves upon the energy-delay-throughput of the previous state-of-the-art TFC mechanism. ETFC provides an extra bypass mode to the VPSR architecture. The three modes of pipeline stages together allow VPSR to dynamically select a configuration to meet either the power budget or the performance requirements. ETFC provides a protocol to improve buffer utilization and enhance the use of tokens. It also incorporates a GEVC
protocol to improve the use of guaranteed tokens. A GEVC guarantees that all the flits belonging to a packet can bypass the token route up to three hops, and is automatically turned off after the tail flit of the packet traverses the route. It is used only when there is a serious traffic congestion. We found that the VPSRDPM scheme can reduce power by $20\% \sim 25\%$ with little performance impact. We also experimented with PARSEC benchmarks, and found that network power is reduced by $19.67\%$, on an average.

6.2 Future Directions

Several extensions are possible to the work presented in this thesis, as discussed next.

6.2.1 Global knowledge based routing

In high-performance computing, optimizing the path between the source and destination nodes is a difficult problem. When upwards of a thousand processing cores are present on an IC, the nodes could be connected in a $32 \times 32$ mesh or in a 3D configuration. In such a network, different regions are likely to have very different traffic patterns. Deterministic routing algorithms, such as DOR, could be very inefficient because of the uncertain traffic patterns encountered, network congestion, and unbalanced resource allocation. In addition, in heterogeneous systems, different network components may have different bandwidths as well as different data processing capabilities. Global knowledge, which provides each processing node with a general idea of which node/region to forward data to, may, therefore, be crucial when optimizing system performance. Current NoC optimization techniques, such as dynamic routing, EVC, TFC, ETFC, VPSR, focus only on local traffic conditions. These techniques are excellent at handling small networks and are able to improve network performance by up to $30\% \sim 50\%$. However, when the network size is increased
to the scale of hundreds or thousands of cores, these techniques may face scalability limitations. In addition, with the emerging promise of photonic on-chip networks and RF communications, multiple means of transmissions may be available, not just electrical communications. Each router, as a result, may have a number of choices for transmitting packets. The combination of global knowledge with local dynamic routing techniques may thus be promising. A hybrid method may be feasible: global routing to route the packet to the destination region and dynamic routing for local packet communications.

Global knowledge based routing is similar to the concept of routing based on a routing table. Each router keeps track of the best path for sending a packet to a region. Initially, there is no such knowledge. Hence, any heuristic algorithm, such as a greedy algorithm, could be used. Each router updates the best route to each region. This information could be implemented with the help of piggybacking or broadcasting. Each node could periodically perform a handshake with its neighboring nodes to exchange routing tables.

Both homogeneous systems and heterogeneous systems can be simulated. The simulator can be made compatible with GEM5. Global knowledge based routing can be implemented as a separate module. The simplest version could be based on just a routing table. A more sophisticated version may rely on the concept of machine learning. The concept is also applicable to other areas, such as power optimization and resource allocation in an on-chip FPGA.

6.2.2 McPAT-PVT

McPAT-PVT is a framework that can be developed for modeling delay, power, and PVT variations for FinFET-based processors. McPAT-PVT can be enhanced from McPAT [33] with several new features:
• It can be based on 2D TCAD FinFET models that capture the simulation accuracy of 3D TCAD models.

• It can support all of the FinFET design styles (SG, IG, and ASG modes).

• Its cache and NoC models can be based on FinCANON.

• It should contain PVT variation models that can be applied to various processor components.

• It should contain macromodels for various processor functional blocks.

• It should enable yield analysis.

McPAT-PVT can be developed on top of the FinCANON framework. The PVT variation models and the FinFET design library can be directly applied to the functional blocks in a processor. The functional blocks in a processor, e.g., arithmetic-logic unit (ALU), multiplication unit (MLU), floating point unit (FPU), memory control unit (MCU), trap logic unit (TLU), flash controller (FLASHC), network interface unit (NIU), peripheral component interconnect bus (PCIB), etc., can be analyzed by FinPrin and represented by macromodels. FinPrin is a tool for analysis of delay and power of FinFET-based circuits. McPAT should be programmed in a hierarchical fashion, as shown in Fig. 4.2. It should be programmed in a modular, object-oriented fashion, such that new processor components can be added to the framework easily. In addition to delay and power analysis of processor components, McPAT-PVT should support yield analysis. Given a processor architecture and its target operation frequency, this yield analysis will derive die yield and lead to optimization strategies to improve it statistically.

The most important part of McPAT-PVT will be the macromodels developed for the processor functional blocks. In the original McPAT, delay and power are implemented as empirical numbers derived from existing processors with scaling factors
for different technologies and processor configurations. However, scaling the delay, power, and area numbers of ALUs and FPUs from those of existing processors does not guarantee correctness for different processor configurations and technology nodes. In McPAT-PVT, the macromodels can be derived based on synthesis results of Design Compiler [166] and FinPrin [88]. Fig. 6.1 shows the macromodel construction flow for the processor functional blocks. It consists of several steps:

1. Synthesize RTL descriptions of the processor components to gate-level Verilog code. Each functional block can be synthesized several times for different frequencies.

2. Translate gate-level Verilog code to the readable format for FinPrin (*.bench files), and analyze the block under different temperatures.
3. Build the macromodels based on the simulated results from FinPrin.

4. Incorporate the generated macromodels in the McPAT-PVT framework to obtain delay/power distribution of the functional blocks.
Bibliography


