NORTHWESTERN UNIVERSITY

Power-Aware and Temperature-Aware Design Automation

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Electrical Engineering and Computer Science

By

Zhenyu Gu

EVANSTON, ILLINOIS

December 2007

ABSTRACT

Power-Aware and Temperature-Aware Design Automation

Zhenyu Gu

This dissertation presents several topics which are related to temperature-aware and power-aware design.

An efficient unified incremental high-level and physical-level synthesis algorithm was presented in Chapter 2, which enable the tight integration between high-level and physical-level design. Chapter 3 presented a temperature-aware high-level synthesis algorithm built upon the framework of Chapter 2. Temperature variations and hot spots account for reliability issue and require more conservative timing margins, thereby reducing performance. Therefore, reliability-aware synthesis flow was presented in Chapter 4. Recent developments in nanoscale devices open new alternatives for low-power embedded system design. Among these, single-electron tunneling transistors hold the promise of achieving the lowest power consumption. Unfortunately, most analysis of SETs has focused on single devices instead of architectures, making it difficult to determine whether they are appropriate for low-power embedded systems. Therefore, a fault-tolerant hybrid SET/CMOS reconfigurable architecture was presented

in Chapter 5. 3D CMOS technology has been developed to overcome the interconnect bottlenecks of 2D design, but at the cost of serious thermal problems due to significant increase of power density by stacking multiple active device layer together. Hence, in Chapter 6, 3D CMP framework was presented and temperature-aware management policy was evaluated built upon this framework.

To Tian

Acknowledgements

First, I would like to thank my advisor, Robert Dick. We closely collaborated on all the work presented in the body of this dissertation. He has all the traits of an excellent research advisor; he is intelligent, diligent, methodical, careful, imaginative, and even-tempered. I appreciate the corrections and suggestions offered by my dissertation readers: Robert Dick, Li Shang, Seda Ogrenci Memik, Hai Zhou, and Russ Joseph.

I was glad to have the opportunity to discuss research with Changyun Zhu, Jia Wang, Yonghong Yang, Ke Meng, and Lin Zhong.

Lastly, and most importantly, I am deeply indebted to my parents and wife for their selfless love, care, support and belief. Without them, I could never have gotten to where I am today.

Zhenyu Gu

Northwestern University

December 2007

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

In this chapter, we give an overview of the system-level/high-level design challenges resulting from the aggressive process scaling. Then, a thermal-aware design flow is proposed to prevent thermal problems. Finally, the structure of this dissertation is introduced.

1.1. Technology Scaling

Today, electronic circuits have developed to extremely high-performance ultra-large-scaled integrated circuits made of CMOS transistors. They have been highly involved into our daily life in the form of internet, cellphones, personal computers, digital cameras, cars (filled with microcontrollers) etc. Moreover, with the development of deep sub-micro devices, the importance of the integrated circuits will be expected to further increase in the future.

This significant progress is mainly driven by the aggressive process scaling in the feature size of VLSI circuits. The main goal of process scaling is to decrease gate capacitance and density, which results in the increase of circuit operation speed and decrease of its power consumption. Moreover, reduction in transistor size increases the number of transistors in the VLSI circuit (about billion transistors nowadays), and enhances parallel operation capability, which leads to further increase in the circuit performance. Today, feature size has been aggressively scaled to 65 nm from 350 nm in 1996, and it has been predicted by International Technology Roadmap for Semiconductors (ITRS) [**2**] that the feature size will continue to shrink to about 32 nm in Year 2010. Furthermore, in the research level, normal transistor of a 5 nm gate length

p-channel MOSFET has been already reported [**3**]. This is approaching to the ultimate limit of the process scaling, which is the distance of atoms in silicon crystals (about 0.3 nm). However, before reaching the ultimate limit of the atomic size, the process scaling will meet a practical limit caused by one of the integration issues: 1) the performance of the chip; 2) the cost of manufacturing; 3) the increase of chip power and heat generation; 4) the reliability and yield degradation of the chip.

Assume that process scaling trends continue (i.e., frequency doubles, supply voltage scales down 30%, active capacitance grows 30%, and die size grows 25%) [**4**], then the power consumption will increase to 2000 W in Year 2010. Even worse, it could reach approximately 10,000 W if the supply voltage process scaling rate is much slower due to its physical limitation [**5**] and leakage power continues to increase. Therefore, it appears that power delivery and dissipation will be primary limiters of performance and integration. Hence, it is really important to propose a power-aware design flow.

Thermal problems can also occur since energy consumed by the transistors is converted into heat. Therefore, increasing IC power consumption raises average and peak temperatures. Temperature variations and hot spots account significantly for electronic failures [**6**], most of which are due to electromigration, hot carrier effects, thermal stress, and oxide thermal breakdown. Power and temperature variation can also lead to significant timing uncertainty, requiring more conservative timing margins, thereby reducing performance. Designers must frequently trade off other design metrics, such as performance, area, and cooling costs, to meet tight temperature constraints. The interaction of power and temperature constraints with other design metrics further increases system complexity. As projected by the ITRS [**2**], further process scaling will be bounded by power consumption and heat dissipation below 65 nm: it is critical to address

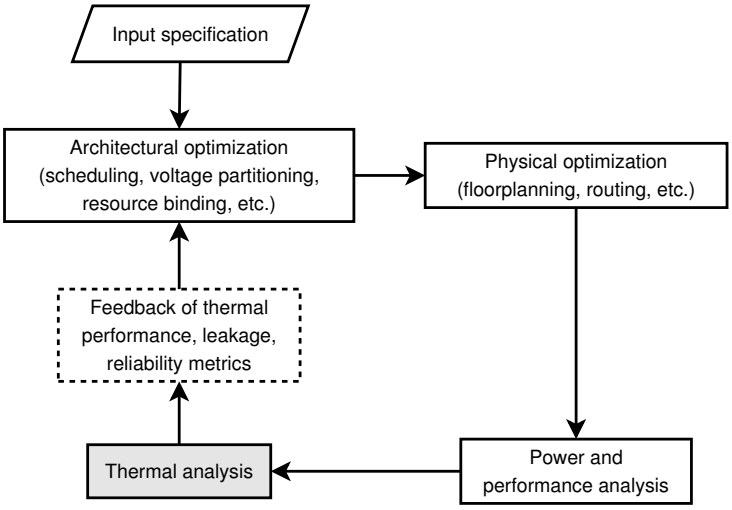


Figure 1.1. Thermal-aware synthesis flow.

the energy and thermal issues during on-chip system design to meet the urgent need of the semiconductor industry and enable future technology scaling.

Thermal-aware design is very challenging. Thermal problems cannot be well solved at any single level of the design process. First, thermal characterization requires detailed physical information, including an IC floorplan and power profile as well as interconnect and chip-package thermal models. Second, thermal optimization requires a unified system-level/high-level and physical-level design flow. At the architectural level, power-aware techniques (such as voltage partitioning, resource allocation and assignment) can reduce IC power consumption, hence the temperature. At the physical level, efficient floorplanning is critical to correctly implement temperature-aware design changes made by high-level design flow while maintaining other design metrics, such as performance, chip area, and cooling cost. Therefore, thermal-aware design requires a comprehensive high-level and physical-level infrastructure.

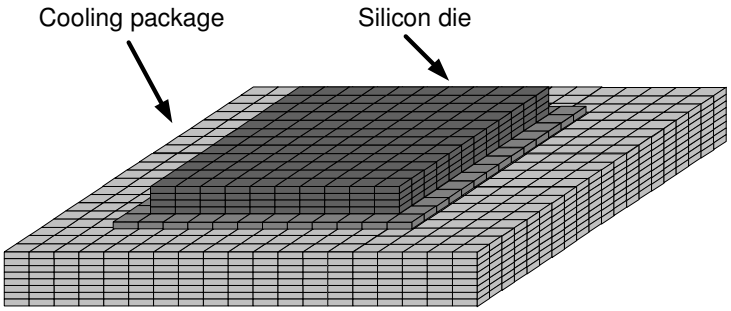


Figure 1.2. Silicon chip and package.

1.2. Power-Aware and Temperature-Aware Design Flow

In this section, we explain the challenges for preventing thermal problems based on our proposed temperature-aware synthesis design flow.

Figure 1.1 shows an integrated behavioral-level and physical-level IC synthesis system [**7**]. This synthesis system uses a simulated annealing algorithm to jointly optimize several design metrics, including performance, area, power consumption, and peak IC temperature. It conducts both behavioral-level and physical-level stochastic optimization moves, including scheduling, voltage assignment, resource binding, floorplanning, etc. Thermal analysis algorithms are invoked to guide optimization moves. An intermediate solution is generated after each optimization move. A detailed two-dimensional active layer power profile is then reported based on the physical floorplan. The power profile and floorplanning information will be used to generate a full chip-package temperature profile as shown in Figure 1.2.

From above design flow, several interesting problems need to be solved in order to provide an efficient thermal-aware design. First, since physical information is necessary to evaluate every high-level decision, a fast unified high-level and physical-level design flow is needed. Second, efficient power minimization techniques are necessary because of strong relationship

between temperature and power consumption. Third, a high-quality floorplanner is important to understand and correctly implement high-level decisions. Finally, accurate thermal analysis algorithm is another performance bottleneck for thermal-aware design flow.

1.3. Dissertation overview

In this thesis, we present the results [**8**,**7**,**9**,**10**,**11**,**12**] of the proposed temperature-aware design flow for both system-level and high-level design system. The rest of this thesis is organized as follows.

First, achieving design closure is one of the biggest challenges for modern VLSI designers. This problem is exacerbated by the lack of high-level design automation tools that consider the increasingly important impact of physical features, such as interconnect, on integrated circuit area, performance, and power consumption. Using physical information to guide decisions in the behavioral-level stage of system design is essential to solve this problem. However, it is unstable and time-consuming to loosely coupled floorplanners for physical estimation with high-level synthesis especially for larger problem instances. Therefore, in Chapter 2, a fast unified high-level and physical-level synthesis system was proposed which has the benefit of efficiency, stability, and better quality results.

Second, thermal effects are becoming increasingly important during integrated circuit design. It is necessary to consider thermal effects during all levels of the design process, from the architectural level to the physical level. Hence, a temperature-aware high-level synthesis system is proposed in Chapter 3. It uses a tightly integrated thermal model and incremental floorplanner to optimize ICs peak temperatures, area, and power consumptions, while meeting performance constraints.

Third, aggressive scaling of CMOS process technology poses serious challenges to the lifetime reliability of ICs. Reduction of fabrication feature sizes and increases in power density have resulted in increasing chip temperature and failure rates. Increasing system integration using these vulnerable devices and interconnects results in reduced system reliability. The severities of many reliability problems, such as time-dependent dielectric breakdown in MOS transistors and electromigration in interconnects, increase exponentially with temperature. Lifetime reliability is becoming an important quality metric in high-performance ICs [**2**]. Optimizing lifetime reliability requires careful planning during IC design and synthesis. Therefore, Chapter 4 presents a comprehensive solution to the reliable multiprocessor system-on-chip (MPSoC) synthesis problem.

Fourth, minimizing power consumption is vitally important in embedded system design since power consumption determines battery lifespan. Ultra-low-power designs may even permit embedded systems to operate without batteries, e.g., by scavenging energy from the environment. Moreover, managing power dissipation is now a key factor in integrated circuit packaging and cooling. As a result, embedded system price, size, weight, and reliability are all strongly dependent on power dissipation. Recent developments in nanoscale devices open new alternatives for low-power embedded system design. Among these, single-electron tunneling transistors (SETs) hold the promise of achieving the lowest power consumption. Unfortunately, most analysis of SETs has focused on single devices instead of architectures, making it difficult to determine whether they are appropriate for low-power embedded systems. Hence, a reliable hybrid SET/CMOS reconfigurable architecture was proposed in Chapter 5, which is the first step in determining the potential of ultra-low-power embedded system design using SET.

In addition, technology scaling has been a major driving force in high-performance microprocessor design. A single integrated circuit (IC) can now incorporate billions of transistors. Therefore, high-performance microprocessor design is moving towards highly-scalable multicore architectures [**13**, **14**, **15**, **16**, **17**, **18**, **19**]. However, interconnect performance has not kept pace with transistor performance. In particular, global interconnects have become major performance bottlenecks. Researchers have proposed several techniques such as repeater insertion, wave pipelining, and multicycling to improve interconnect performance. However, they remain performance limiters and have substantial power consumption. Three-dimensional integrated circuits (3D ICs) [**20**, **21**, **22**, **23**] are very promising approaches for improving interconnect delay and power consumption by dramatically reducing interconnect length. Multiple device layers are stacked and electrically connected with vertical interconnects, i.e., interwafer vias. This comes at the cost of serious thermal problems due to the significant increase of power density by stacking multiple device layers together. Therefore, in Chapter 6, a 3D chip multiprocessor (CMP) simulation framework was built and thermal-aware management policy was proposed and evaluated.

Finally, we will give a conclusion of this dissertation and the potential future research problems are proposed in Chapter 7.

CHAPTER 2

Unified Incremental High-Level and Physical-Level Synthesis

Achieving design closure is one of the biggest challenges for modern VLSI designers. This problem is exacerbated by the lack of high-level design automation tools that consider the increasingly important impact of physical features, such as interconnect, on integrated circuit area, performance, and power consumption. Using physical information to guide decisions in the behavioral-level stage of system design is essential to solve this problem. In this chapter, we present an incremental floorplanning high-level synthesis system. This system integrates high-level and physical design algorithms to concurrently improve a design's schedule, resource binding, and floorplan, thereby allowing the incremental exploration of the combined behavioral-level and physical-level design space. Compared with previous approaches that repeatedly call loosely coupled floorplanners for physical estimation, this approach has the benefits of efficiency, stability, and better quality of results. The average CPU time speedup resulting from unifying incremental physical-level and high-level synthesis was 24.72× and area improvement was 13.76%. The low power consumption of a state-of-the-art, low-power, interconnect-aware high-level synthesis algorithm was maintained. The benefits of concurrent behavioral-level and physical design optimization increased for larger problem instances.

2.1. Introduction

Process scaling has enabled the production of integrated circuits (ICs) with millions of transistors. This has allowed the design of more full-featured and high-performance ICs. However,

these increased capabilities have come at a cost. In order to deal with increased design complexity and size, it is becoming increasingly important to automate the higher levels of the design process.

High-level synthesis systems [**24**, **25**, **26**, **27**] automatically convert behavioral, algorithmic, descriptions of design requirements, e.g., control data flow graphs (CDFG) [**28**], into optimized register-transfer level (RTL) descriptions in languages such as VHDL or Verilog. Based on a behavioral description, a high-level synthesis system determines an allocation of resources, assignment of operations to resources, and a schedule for operations, in an attempt to satisfy the design specifications and minimize some combination of delay, area, and power consumption [**29**, **30**, **31**, **32**, **33**, **34**, **35**, **36**, **37**, **27**, **38**, **39**]. Recently, in order to improve design area or performance estimation, a number of researchers have considered the impact of physical details, e.g., floorplanning information, on high-level synthesis [**40**, **41**, **42**, **43**, **44**, **45**].

In the past, it was possible for high-level synthesis algorithms to focus on logic, i.e., functional units such as adders and multipliers. The contribution of wire delay and area was typically neglected without much loss of accuracy. Focusing on logic was once reasonable since logic was responsible for the majority of delay and power consumption. However, process scaling into the deep sub-micron realm has changed the focus of VLSI design from transistors to global interconnect. It is no longer possible to simplify the high-level synthesis problem by ignoring interconnect.

Taking interconnect cost into consideration during high-level synthesis has attracted significant attention. In previous work [**46**, **47**, **48**, **49**, **50**], the number of interconnects or multiplexers was used to estimate interconnect cost. The performance and power impacts of interconnect and interconnect buffers are now first-order timing and power considerations in VLSI design [**51**].

It is no longer possible to accurately predict the power consumption and performance of a design without first knowing enough about its floorplan to predict the structure of its interconnect. This change has dramatically complicated both design and synthesis. For this reason, a number of researchers have worked on interconnect-aware high-level synthesis algorithms [**52**, **53**, **54**]. These approaches typically use a loosely coupled independent floorplanner for physical estimation. Although this technique improved on previous work by allowing estimation of physical properties, there are two drawbacks for this approach. First, the independent floorplanner may not be stable, i.e., a small change in the input netlist may result in a totally different floorplan. A *move* is a discrete change made to a solution during optimization that results in transition to a new position in the solution space. Floorplan instability may result in a high-level synthesis algorithm that bases its moves on cost functions without continuity. Second, even if a floorplanner is stable, creating a floorplan from scratch for each high-level synthesis move is inefficient, given the fact that the new floorplan frequently has only small differences with the previous one. The constructive approach works for small problem instances but is unlikely to scale to large designs. New techniques for tightly coupling behavioral and physical synthesis that dramatically improve their combined performance and quality are now necessary.

Incremental automated design promises to build tighter relationship between high-level synthesis and physical design, improving the quality of each [**55**, **56**, **8**]. A number of high-level synthesis algorithms are based on incremental optimization and are, therefore, amenable to integration with incremental physical design algorithms. This has the potential of improving both quality and performance. Incremental methods improve quality of results by maintaining important physical-level properties across consecutive physical estimations during synthesis.

Moreover, they shorten CPU time by reusing and building upon previous high-quality physical design solutions that required a huge amount of effort to produce.

This chapter describes an incremental high-level synthesis system that reduces synthesis time dramatically while producing ICs with better area and low power consumption compared to a state-of-the-art power-aware high-level synthesis algorithm. The benefits of this approach increase with increasing problem size and complexity. Our work is based on the interconnect-aware high-level synthesis tool, ISCALP [**53**], which was based on the low-power datapath synthesis tool, SCALP [**27**]. We reuse the power modeling and iterative-improvement high-level synthesis framework from ISCALP. However, this work differs from previous work in that a truly incremental floorplanner is used to estimate the interconnect structure [**57**], instead of a fast constructive algorithm. Moreover, the high-level synthesis algorithm, itself, is made incremental. As shown in Section 2.5, this resulted in an average speedup of 24.72$\times$ and an average area improvement of 13.76%, while maintaining the low power consumption of a state-of-the-art power-aware high-level synthesis algorithm. In addition, wire delay is considered in this work to guarantee that the implementation meets its performance requirements.

This chapter is organized as follows. Section 2.2 explains the motivation for this work. Section 2.3 describes the design flow for the proposed high-level synthesis system. The details of our incremental floorplanner are introduced in Section 2.4. Experimental results are presented in Section 2.5. Conclusions and future work are presented in Section 2.6.

2.2. Motivation

In this section, we first present definitions useful in the discussion of high-level synthesis. Then motivational examples are given based on our observations of the synthesis process.

Examples are used to explain and motivate the use of unified high-level and physical-level optimization.

2.2.1. Definition

The input to ISCALP is a control-data flow graph (CDFG), G, an input arrival (and output sampling) period, T_s, and a library, L, of functional units for data path implementation. ISCALP produces an RTL circuit in which power consumption (including logic and wire power consumption) and estimated area are optimized. The ISCALP algorithm has two loops. Given the supply voltage, the outer loop incrementally reduces the number of control steps, *csteps*, from its maximum to minimum value, where $csteps$ is defined as

$$csteps = T_s \times f \tag{2.1}$$

or alternatively,

$$T_{clock} = T_s / csteps \tag{2.2}$$

In the above equations, each control step corresponds to a time period of one clock cycle, and the sample period T_s is the constraint on the input data rate. The solution got from high-level synthesis must be able to process an input sample before the next one arrives. For a given specification, the sample period is fixed. Hence, *csteps* indicates the number of clock cycles required to process an input sample. The variable f is the chip clock frequency. T_{clock} is the chip clock period. ISCALP searches for the lowest-power architecture for each possible value of *csteps*.

Given a value of *csteps*, which allows the clock period to be determined, the inner loop first uses the fastest available functional unit from the library to implement each operation. An as-soon-as-possible (ASAP) schedule is then generated for the initial solution to determine whether it meets its timing requirements. The initial solution is then further optimized. Having obtained an initial solution that meets the sample period constraint for the current value of *csteps*, the iterative improvement phase attempts to improve the architecture by reducing the switched capacitance while satisfying the sample period constraints. More details can be found in literature [**27**, **53**].

2.2.2. Motivating Example

ISCALP employs a fast constructive slicing floorplanner based on netlist partitioning and rotation/orientation selection to obtain a floorplan optimized for wire length and area [**58**, **59**]. Although it improved on its predecessors by considering the impact of floorplanning on synthesis, there are several drawbacks to this approach.

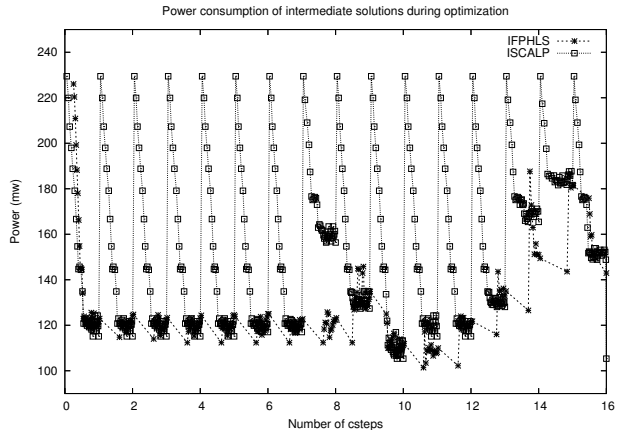
First, an incremental high-level synthesis algorithm only changes a small portion of the modules and connections in each move. However, a constructive (conventional) floorplanner always starts floorplanning from scratch. It is not efficient because it does not reuse the floorplan information obtained during the previous run. Moreover, it is possible that the newly-produced floorplan will be totally different from the previous one, despite only small changes in the set of modules and interconnections. This lack of autocorrelation in floorplan solutions may result in a high-level synthesis algorithm basing its future moves on information that immediately becomes invalid after the moves are applied.

Second, the efficiency of the constructive slicing structure floorplanner decreases dramatically for blocks with non-unity aspect ratios (ISCALP assume blocks with unity aspect ratio). As a result of constraining the solution space to slicing floorplans, it is prone to reaching sub-optimal solutions. However, simply replacing the slicing floorplanner with a high-quality floorplanner would result in unacceptably-high CPU time.

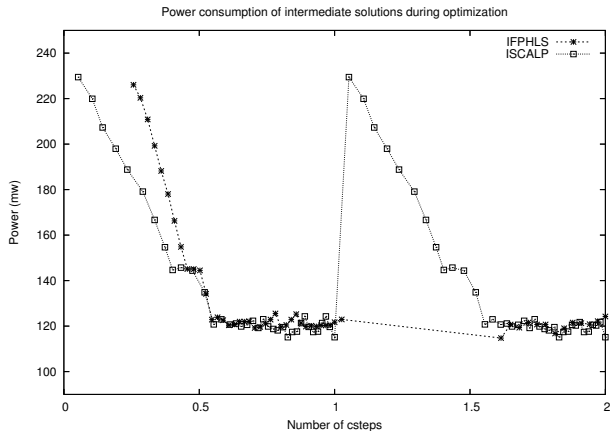
To solve these problems, we propose an incremental iterative improvement high-level synthesis algorithm tightly integrated with a high-quality incremental floorplanner. This synthesis system is called IFP-HLS, i.e., incremental floorplanning high-level synthesis. We run the same benchmarks on both ISCALP and IFP-HLS, listing the number of merge operations and CPU time for each benchmark in Table 2.1 and Table 2.2.

$$t_{total} = N_{moves} * (t_{HLS} + t_{fp}) \tag{2.3}$$

As Equation 2.3 shows, the CPU time of the high-level synthesis run can be divided into two parts: high-level synthesis moves and the resulting physical design carried out by the floorplanner. As shown in Table 2.3, floorplanning is the most time consuming of these. It uses at least 75.69% of the CPU time on average for both ISCALP and IFP-HLS. As shown in Table 2.1 and Table 2.2, IFP-HLS achieves an average reduction of 50% in the number of high-level synthesis merge operations compared to ISCALP. This results in a large reduction in floorplanner CPU time. The reduction in moves, and CPU time, is mainly due to the incremental high-level synthesis and floorplanning algorithms used in IFP-HLS. Many high-level synthesis moves result in time-consuming changes to the floorplan. IFP-HLS can greatly reduce CPU time by reducing



(a) Power consumption of intermediate solution during optimization for all the values of csteps.



(b) Power consumption of intermediate solution during optimization for the values of csteps from 0 to 2.

Figure 2.1. Power consumption of intermediate solution during optimization of ISCALP & IFP-HLS.

the number of merge operations, especially for larger benchmarks which have bigger solution space to explore.

Figure 2.1 illustrates the power consumptions of intermediate solutions during optimization in ISCALP and IFP-HLS. For each value of csteps, we plot the intermediate solutions produced by the optimization algorithm. Note that these intermediate solutions all have the same value of

csteps. Incremental optimization allows IFP-HLS to focus on the most promising (low-power) regions of the solution space while ISCALP must essentially restart optimization for each new potential clock frequency. This allows improvement to both optimization time and solution quality for IFP-HLS.

Note that ISCALP starts the floorplanner from scratch after each high-level design change. The incremental physical and architectural optimization used in IFP-HLS reduces CPU time dramatically, especially for large applications. Table 2.1 and Table 2.2 indicates that the average CPU time speedup is 24.72×. The improvement is greatest for the largest benchmarks. For example, when run on Random300, ISCALP does not finish within 5 days, while IFP-HLS finishes within 4 hours. In addition, IFP-HLS achieves 13.76% improvement in area compared to ISCALP.

The above examples clearly illustrate the value of using unified incremental physical-level and high-level design synthesis. As shown in detail in Section 2.5, this approach improves both design quality and CPU time.

2.3. Incremental High-Level Synthesis

In this section, we describe our incremental floorplanning high-level synthesis algorithm (IFP-HLS). IFP-HLS is built upon ISCALP [**53**]. However, incorporating incremental floorplanning required substantial changes to that algorithm, resulting in a new low-power incremental floorplanning high-level synthesis algorithm. IFP-HLS considers both datapath and interconnect power consumption. As shown in Figure 2.2, the CDFG is simulated with typical trace in order to profile the switching activity of each operation and data transfer edge. A

RTL design library is set up to provide the power and area information. The profiling information combined with this RTL design library and floorplanner is then used to evaluate the power consumption of both datapath and interconnect. IFP-HLS uses a new incremental method for improving functional unit binding during high-level synthesis. Although this improvement, alone, would result in a reduction in synthesis time, its motivation was to facilitate the integration of an incremental floorplanning algorithm with high-level synthesis in order to improve solution quality and reduce synthesis time. This allows the high-level synthesis algorithm to determine the physical position of each module during optimization, enabling interconnect power consumption and delay estimation.

2.3.1. Incremental High-level Synthesis Framework

In this section, we describe our incremental high-level synthesis tool, IFP-HLS. The flowchart of IFP-HLS is shown in Figure 2.2. IFP-HLS differs from ISCALP in a number of ways. Instead of generating an initial solution for each value of *csteps*, IFP-HLS only generates one solution at the maximum value of *csteps* and incrementally changes the solution as *csteps* decreases. Thus, in addition to using incremental floorplanning, IFP-HLS also eliminates redundant moves by taking advantages of incremental steps in high-level synthesis. Initially, we still use an ASAP schedule and fully parallel allocation to estimate whether there exists a valid solution for the current value of *csteps*. If not, it is not necessary to do further moves for the current number of control steps because a binding that further reduces the finish time of an ASAP schedule is not possible. However, if an ASAP schedule meeting the timing requirements is possible, we will use the best solution from the previous value of *csteps* and reschedule it based on the current value, which is equal to the previous *csteps* minus 1. If, after rescheduling, the

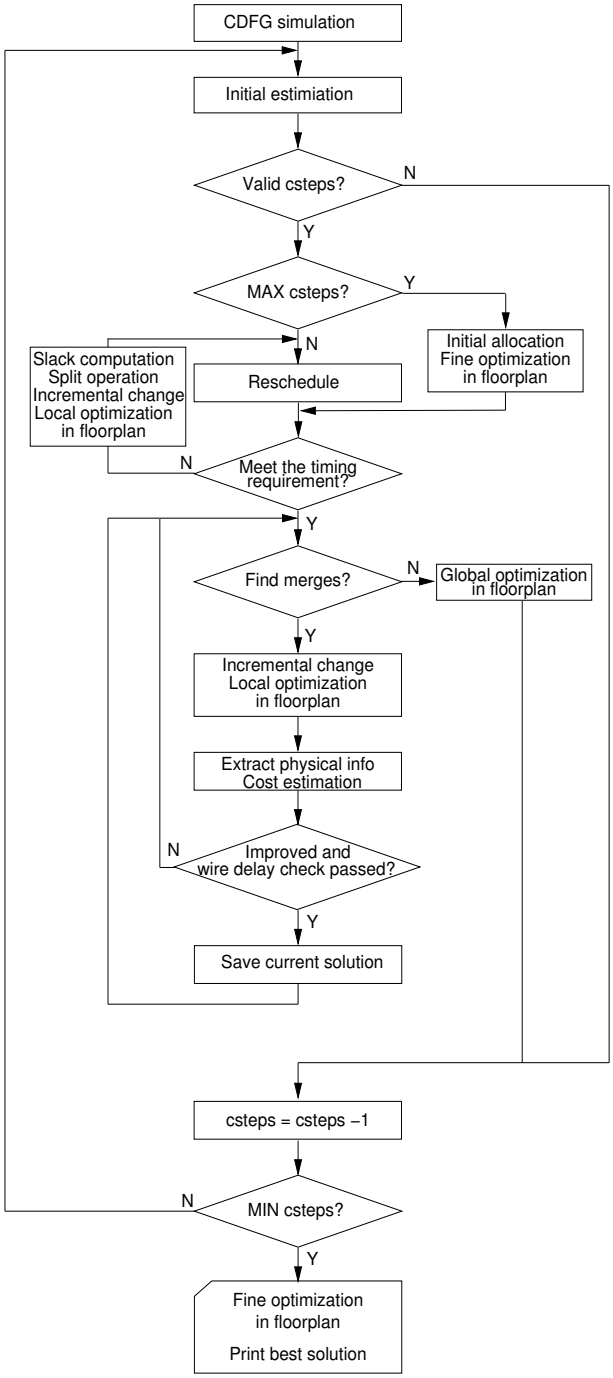


Figure 2.2. Incremental high-level synthesis algorithm.

solution meets its timing requirements, rebinding is not necessary. Otherwise, it will be necessary to parallelize some of the operations to improve performance. The *split move* is used to eliminate resource contention by splitting a pair of operations that were initially assigned to the same functional unit onto separate functional units. A detailed description of the split move may be found in Section 2.3.2.

For a given value of *csteps*, when a move is chosen, IFP-HLS incrementally changes the floorplan to see whether the change improves solution quality. If so, the change is accepted. Otherwise, the change is rejected and other moves are attempted. This technique differs from that in ISCALP. In ISCALP, floorplanning is only done at the end of each *csteps* iteration; it does not take advantage of solution correlation to save effort across *csteps* values. ISCALP uses only power consumption to guide high-level synthesis moves. In contrast, IFP-HLS uses a weighted sum of area and power consumption (pW), with a ratio of 1 μm^2 to 5 pW, in order to evaluate solution quality.

A high-quality incremental floorplanner was developed and incorporated into IFP-HLS to guide high-level synthesis moves. Each time the high-level synthesis algorithm needs physical information, it extracts that information from the current, incrementally generated, floorplan. Costs derived from the floorplan are also used to guide high-level synthesis moves. By using incremental floorplanning, closer interaction between high-level synthesis and physical design is possible, i.e., the high-level synthesis algorithm may determine the impact of potential changes to binding upon physical attributes such as interconnect power consumption and area.

The core idea of incremental design is to maintain good physical-level properties across consecutive physical estimations during high-level synthesis moves. It is possible to apply the idea of using an incremental optimization framework to integrate other algorithms, provided

that the algorithms at each level of design can be made incremental. Let us consider a few other examples. In force-directed scheduling, all the operations may be scheduled iteratively in the order of increasingly cost and the cost of scheduling each unscheduled operation is updated after each operation is scheduled. This provides a potential opportunity to tightly integrate an incremental floorplanner to physical information feedback. For maximal clique based resource sharing, since it is a NP-hard problem, a heuristic algorithm will be used in practice. As long as the heuristic algorithm itself is iterative, it can be made incremental.

In summary, IFP-HLS performs scheduling, allocation, and binding by iteratively changing *csteps* and determining whether operations need to be rescheduled or re-bound (split) in order to meet timing constraints. At each step the floorplan is updated and re-optimized.

2.3.2. Extended Move

This subsection describes the split moves, rescheduling, and a new graph technique to determine split locations.

We observed that when *csteps* decreases by one, each individual operation takes, at most, the same number of control steps as it did for the previous value of *csteps*. Given that *csteps* is no less than the previous *csteps* minus one, we can conclude that the ASAP schedule for the previous value of *csteps* violates the deadline for the current value of *csteps* by, at most, one clock cycle. We will use node i's slack, S_i, to represent this information, which is defined as follows:

$$S_i = LST_i - EST_i \tag{2.4}$$

Here, EST_i is the earliest start time and LST_i is the latest start time which were computed by a topological sort. Hardware resource contention has already been considered.

Nodes with non-negative slack values do not imply timing violations. However, nodes with slack values of -1 cause timing violations, i.e., they must be executed one cycle earlier. These timing violations can be removed by splitting merged operations which, although useful for previous values of *csteps*, now harm performance. Based on this observation, the split move is used to eliminate timing violations. Therefore, the whole high-level synthesis algorithm is implemented in an incremental way from maximum to minimum values of *csteps* without rebinding from scratch at each value of *csteps*. Few changes to binding and scheduling are required as a result of single-unit change to *csteps*. However, in order to meet timing requirements, it is sometimes necessary to split operators mapped to the same functional unit. The split move makes it possible to quickly apply these isolated changes. Previous high-level synthesis systems, e.g., SCALP and ISCALP, started from a fully parallel implementation for each value of *csteps* and repeatedly merged operators to reduce area. Although both techniques are reasonable in the absence of an integrated floorplanner, the incremental approach used in IFP-HLS speeds optimization (without degrading solution quality) by requiring far fewer changes to the floorplan. Used together, the split and merge moves allow complete exploration of the solution space. However, the primary goal of changing the number of control steps is meeting timing constraints. We therefore start our exploration of the solution space at the most promising region by iteratively splitting functional units on the critical timing path.

The reschedule and split procedure is shown in Algorithm 1. We will give an example to further describe this procedure. Consider the data flow graph shown in Figure 2.3(a), in which arrows represent the data dependencies. Scheduling and allocation yield the DFG in

Algorithm 1 Reschedule and Split Procedure

```
1: Reschedule the design
2: Compute slacks of all operations
3: while there exists negative slack do
4:     Compute slack of all operations
5:     Construct graph including all the operations and edges with negative slack
6:     Use maximum flow to find the minimum cut in the graph
7:     Do the split move
8:     Reschedule the design
9: end while
```

Figure 2.3(b). Here, we can see that three functional units (FUs) are used. Operations *1 and *2 share FU1, operations *3, *4, and *5 share FU2, and operation +1, +2, and +3 share FU3. The sample period is 108 ns, each multiplication takes 20 ns, and each addition takes 10 ns. When *csteps* is reduced from 10 to 9, instead of binding and scheduling from scratch, the algorithm reschedules based on current binding. For this example, after rescheduling, there are still timing violations for operations *3, *4, and *5 because they were all bound to FU2, as shown in Figure 2.3(c). Therefore, the split move is necessary in order to allow all operations to meet their timing requirements. A description of the split move follows.

Based on the result of slack computation, we produce a graph including all the operations with negative slack. Each operation is represented by a node. In addition, there are three kinds of edges, defined as follows:

(1) Data dependency edges, indicating that the destination node takes the source node's data as input;

(2) Merge edges, indicating that the two nodes are currently bound to the same functional unit or same storage unit; and

(3) Pseudo edges, used to restructure the graph for application of the min-cut algorithm. A pseudo source node and pseudo sink node are introduced into the graph. All input

nodes are connected to the pseudo source node and all output nodes are connected to the pseudo sink node.

After constructing this graph, a min-cut algorithm is executed. First, an infinite capacity is assigned to all the pseudo edges and data dependency edges. Merge edges are each given capacities of one. If two nodes are connected by both a data dependency edge and a merge edge, the merge edge is eliminated because split moves on nodes sharing dependency edges do not improve the timing properties. Using a min-cut algorithm in this manner splits a minimal cardinality subset of nodes, allowing a reduction in the finish time of the ASAP. One could consider the impact of area and power consumption within the min-cut max-flow algorithm by weighting the edges appropriately. However, this would generally lead to to additional split operations, increasing CPU time.

Although decrementing *csteps* may increase delay by at most one clock cycle, there may be some value of *csteps* for which even fully parallel bindings do not allow an ASAP schedule to meet its timing constraints. Therefore, min-cut and rescheduling may not be carried out for some values of *csteps*. After the split move, the operations are rescheduled and slack is recomputed to determine whether timing constraints are met.

The algorithm described above was used to construct the graph shown in Figure 2.3(d). The dashed lines represent merge edges. The solid lines represent pseudo-edges and data dependence edges. Nodes S and T represent pseudo source and sink nodes respectively. After slack computation, we eliminate all the nodes and edges which are not on the critical path and assign a capacity of one to merge edges and a capacity of infinity to other edges, as shown in Figure 2.3(e). For this example, it is possible to cut through either edge (+2, +1), or edges (*3, *1) and (*4, *2). Here, we cut through +2 and +1, which is the minimal cut, thereby assigning +1

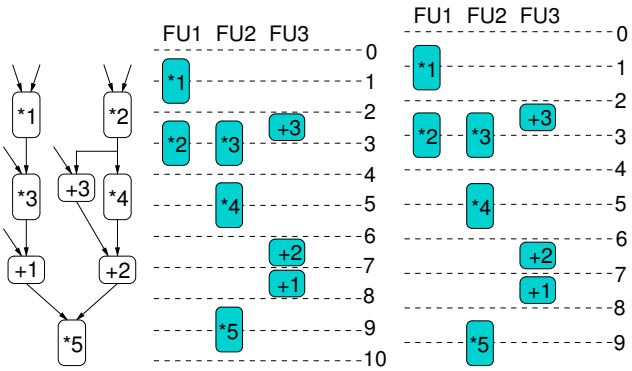
to a new functional unit, FU4. +3 and +2 remain bound to the original functional unit, FU3. As shown in Figure 2.3(f), no operation violates timing constraints.

Another case must also be considered. If no valid solutions exist for the current value of *csteps*, IFP-HLS will skip further optimization and decrement *csteps*. IFP-HLS may reach a valid value of *csteps* after repeatedly decrementing *csteps*. In this case, the slack values for some operations may less than -1. Hence, the value of *csteps* is decremented and the split move, followed by rescheduling, are repeated until a valid solution is produced. This process is as shown in Figure 2.4.

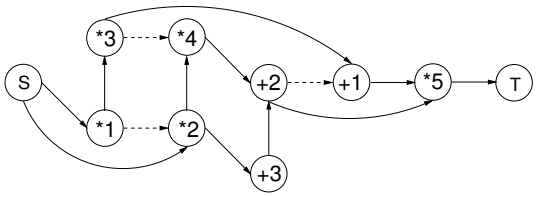
2.4. Incremental Floorplanning

As discussed in previous sections, in order to introduce incremental combined behavioral and physical optimization into high-level synthesis, a high-quality incremental floorplanner is necessary. We have tested this idea by building an incremental simulated annealing floorplanner into the IFP-HLS algorithm. In this section, we describe this incremental floorplanner.

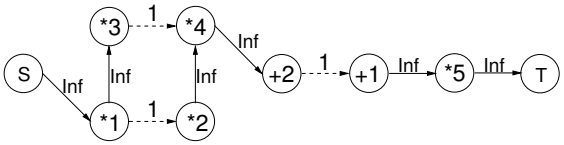
This floorplanner handles blocks with different aspect ratios and produces non-slicing floorplans. Unlike the netlist partitioning approach used in ISCALP, it was designed primarily for quality, not speed. Although the impact on synthesis time would prevent incorporation of a conventional high-quality floorplanner in the inner loop of a high-level synthesis system, using incremental floorplanning enables both high quality and low synthesis time. High-level synthesis moves typically remove a single module or split a module into two. Therefore, many changes are small and their effects on the floorplan are mostly local. We reuse the modified previous floorplan as a starting point for each new floorplan. The previous floorplan is optimized. Therefore, re-optimization of the current floorplan to incorporate local changes is fast.



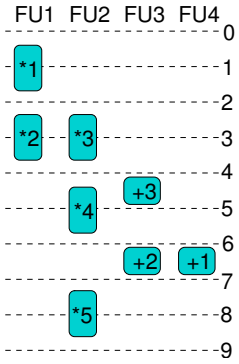
(a) CDFG (b) Scheduling and allocation. (c) Timing violation.



(d) Slack computation.



(e) Min-cut flow.



(f) After split move.

Figure 2.3. Incremental changes on HLS.

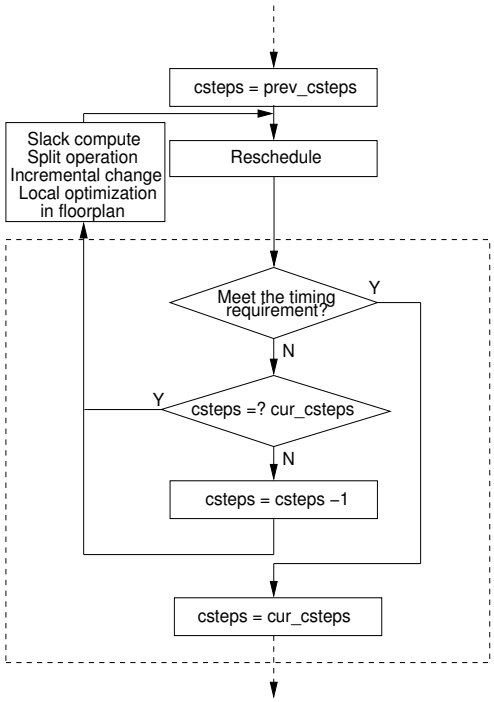


Figure 2.4. Iterative split move for slack smaller than -1.

In practice, we have found that this technique leads to quality-of-results and performance improvements over constructive floorplanning, even when compared with a very fast constructive floorplanner.

2.4.1. Floorplan Representation

The Adjacent Constraint Graph (ACG) floorplan representation is used within IFP-HLS's incremental floorplanner [**57**, **60**, **61**]. This representation will be briefly summarized here.

An ACG is a constraint graph satisfying the following three conditions: first, there is at most one relation (either horizontal or vertical) between any pair of vertices; second, there are no transitive edges; third, there are no *crosses*. A cross is a special edge configuration that can result in quadratic number of edges in the constraint graph. Figure 2.5 shows two cases of

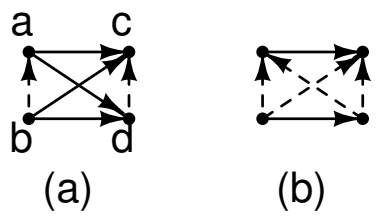


Figure 2.5. (a) Horizontal cross and (b) vertical cross.

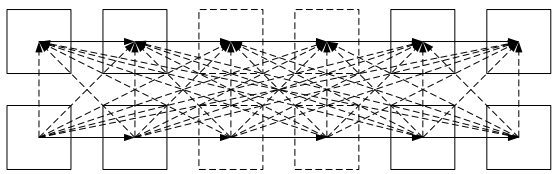


Figure 2.6. A constraint graph without over-specifications and transitive edges can have quadratic number of edges.

crosses and Figure 2.6 shows that a constraint graph with crosses may have quadratic number of edges even when the first two conditions are met. It is proved that the number of edges in an ACG is at most $O\left(n^{1.5}\right)$ where n is the number of vertices [**61**].

The operations of removing and inserting vertices in an existing ACG are designed to reflect high-level binding decisions, i.e., merging and splitting. To obtain the physical position of each module, packing based on longest path computation is employed since the ACG itself is a constraint graph.

Perturbations on the graph are designed so that the ACG can be used in an iterative optimization heuristic such as simulated annealing (SA). They change the graph topology locally and have straightforward meanings in physical space. Since the interconnect lengths are determined by the physical positions of modules, which in turn depend on the graph topology,

applying these perturbations changes the interconnects locally. Other perturbations include rotating a module and exchanging the two modules represented by any pair of vertices. The latter changes the interconnect globally.

2.4.2. Incremental Floorplanner

There are four situations in which the incremental floorplanner is called by the IFP-HLS framework. First, a floorplan should be generated after each ASAP schedule is produced. We call this an *initial* floorplanning. Second, a floorplan should be modified and optimized after each high-level synthesis move. We call this *per-move* floorplanning. Third, for each *csteps* value, a floorplan for the best binding should be generated and compared to the existing best floorplans. We call this *per-cstep* floorplanning. Fourth, after determining the best clock frequency and binding, floorplanning is carried out to provide the final result. We call this *final* floorplanning.

Although initial, per-cstep, and final floorplanning are done with simulated annealing for quality, per-move floorplanning requires fewer global changes and less hill climbing. Moreover, perturbations resulting from high temperatures may disrupt high-quality floorplan structures. Therefore, it is reasonable to use lower temperatures for per-move floorplanning. In practice, we have found that using a temperature of zero results in good quality and performance. In other words, although simulated annealing is necessary in many cases, per-move floorplanning is done with a greedy iterative improvement algorithm.

The details of our approach follow. First, after generating the first ASAP schedule and binding, we have an initial set of modules and interconnections. Simulated annealing is used to obtain an initial floorplan. Since every interconnect net has exactly one driving module,

multi-pin nets are broken into two-pin wires with the driving module as the source. The wire length is calculated as the Manhattan distance between the two modules connected by the wire. At this point, the unit-length switched capacitances of data transfers between two modules are available. We use these as weights for the wire lengths. The weighted total wire length is related to power consumption, i.e., optimizing weighted wire length minimizes interconnect power consumption. A weighted sum of the area and the interconnect power consumption is calculated as the floorplanner's cost function, that is,

$$A + w \sum_{e \in E} C_e D_e \tag{2.5}$$

where A is the area, w is the power consumption weight, E is the set of all wires, e is an interconnect wire, C_e is the unit-length switched capacitance for the data transfer along e, and D_e is the length of e. With this approach, we optimize the floorplan for both the interconnect power consumption and the area. The resulting floorplan will be improved during the consecutive incremental floorplanning high-level synthesis moves. Therefore, the number of simulated annealing iterations is bounded to reduce synthesis time.

After each high-level synthesis move, per-move floorplanning first modifies the previous floorplan by removing or splitting a module. The modules and switched capacitances are updated based upon the impact of these merges and splits. The floorplan is then re-optimized with a greedy iterative improvement algorithm using the same cost function as the simulated annealing algorithm. The greedy improvements are divided into consecutive rounds. In every round we apply the same number of perturbations to the floorplan. If less than 10% of the perturbations result in reduced costs, re-optimization stops. Although it would be easy to use a low simulated annealing temperature to allow some hill climbing during re-optimization, this was

not necessary in practice. It should be pointed out here that changes to switched capacitances may require a few global changes in the ACG to obtain power consumption optimized floorplans. Therefore, we still allow the exchange perturbation to change the floorplan globally, but reduce its frequency to favor local perturbations.

When we find the best binding for a given value of *csteps*, we do per-cstep floorplanning and compare the result with the best floorplan from previous value of *csteps*. This time non-zero temperature simulated annealing is used because it increases floorplan quality. These normal simulated annealing runs occur only once per *csteps* value, allowing their CPU costs to be amortized.

After determining the best binding across all the possible values of *csteps*, a final floorplanning run is carried out for that binding. This final floorplanning occurs only once per synthesis run. Therefore, it is acceptable to use a higher-quality, but slower, annealing schedule than those in the inner loop of high-level synthesis, thereby reducing chip area and interconnect power consumption.

During the annealing schedule, we use a constant multiplicative cooling factor, r, i.e.,

$$T' = r \times T \tag{2.6}$$

where T is the current temperature and T' is the temperature for the next iteration. The cooling factors for initial, per-cstep, and final floorplanning are 0.7, 0.8, and 0.9 respectively. At one temperature, if less than 10% of the perturbations are accepted, the annealing process stops. The ratio between the numbers of the perturbations at one temperature for initial, per-cstep, and final floorplanning is 1 : 1 : 5. The number of perturbations per round for per-move floorplanning is the same as that in the final floorplanning.

The interconnect power consumption weight, w, is computed during synthesis for each floorplanning to avoid the difficulty of determining a proper value for all the situations. Before each floorplanning, we calculate the area-to-power-consumption ratio, w_0, using the existing floorplan, which is either the previous floorplan for per-move, per-cstep, and final floorplanning or the starting floorplan for initial floorplanning. For initial, per-cstep and final floorplanning, the weight w is set to $0.5 \cdot w_0$ to balance the area and the interconnect power consumption. For per-move floorplanning, it is more important to provide a prediction of the trend of interconnect power consumption in a limited time so that w is set to $2.5 \cdot w_0$ instead. Note that in this stage, not area cost but the prediction of the interconnect power consumption is the major consideration. Therefore, the wire length weight was set to be a big value compared to the area weight.

2.5. Experimental Results

In this section, we present the results produced by the IFP-HLS incremental floorplanning high-level synthesis algorithm described in Sections 2.3 and 2.4 when run on a number of benchmarks. The results generated by ISCALP and IFP-HLS are compared. As explained in Section 2.3.1, both approaches optimize area and power consumption. The experiments were conducted on Linux workstations with dual 933 MHz Pentium III processors and 512 MB of random access memory.

2.5.1. Benchmarks

We evaluated seventeen high-level synthesis benchmarks using a 0.18 µm technology library. *Chemical* and *IIR77* are infinite impulse response (IIR) filters used in industry. *DCT_IJPEG*

is the Independent JPEG Group's implementation of digital cosine transform (DCT) [**62**]. *DCT_Wang* is a DCT algorithm named after the inventor [**63**]. Both DCT algorithms work on 8×8 arrays of pixels. *Elliptic*, an elliptic wave filter, comes from the NCSU CBL high-level synthesis benchmark suite [**64**]. *Jacobi* is the Jacobi iterative algorithm for solving a fourth order linear system [**65**]. *WDF* is a finite impulse response (FIR) wave digital filter. The largest benchmark is Jacobi with 24 multiplications, 8 divisions, 8 additions, and 16 subtractions. In addition, we generate five CDFGs using a pseudo random graph generator [**66**]. Random100 has 20 additions, 15 subtractions, and 19 multiplications. Random200 has 39 additions, 44 subtractions, and 36 multiplications. Random300 has 59 additions, 58 subtractions, and 72 multiplications.

IFP-HLS had better performance than ISCALP on these large randomized benchmarks. In order to determine whether the improved performance of IFP-HLS was the result of random graph structure or benchmark size, we generated two structured benchmarks, Small and Serial. Small is composed of five operations connected in parallel. Serial is composed of 45 operations connected in serial. As shown in Table 2.1 and Table 2.2, IFP-HLS has better CPU time for structured the large benchmark Serial. This is consistent with the results for other other large benchmarks.

The area of each benchmark described in this section was estimated using pre-synthesized functional-units (e.g., adders, multipliers, etc.) based on NEC's 0.18 µm process and the floor-planner from high-level synthesis tool. The logic power consumption of each benchmark was evaluated using power models from the pre-synthesized functional-unit level design library. A full-system switching activity simulator was used during power consumption computation.

Wire power consumption and wire delay were calculated based on the wire capacitances estimated using Cong's and Pan's technique [**67**] and the wire length information from floorplanner of high-level synthesis design tools. As described in Section 2.3, both logic and wire delays were calculated to determine whether each design meets its timing requirements. However, since the wire delay estimation is only implemented in IFP-HLS; this function was not used when comparing to ISCALP.

2.5.2. Results

The results of running ISCALP and IFP-HLS on non-unity aspect ratio functional units are shown in Figure 2.7. As shown in the Figure 2.7(a), Figure 2.7(b), Table 2.1, and Table 2.2, IFP-HLS achieves an average CPU time speedup of 24.72×, 13.76% improvement in area, and 50% reduction in the number of merge move in comparison with ISCALP. Low power consumption is maintained.

ISCALP uses a constructive floorplanner that may suffer performance degradation when used with non-unity aspect ratio functional units. In order to determine whether the improvement in quality and run time were the result of the specific type of floorplanner used in ISCALP, we repeated all experiments using only unity aspect ratio functional units. As shown in Figure 2.8, Table 2.1, Table 2.2, and Table 2.4, the IFP-HLS algorithm achieves an average CPU time speedup of 2.03×, 11.32% improvement in area, and 54% reduction in the number of merge move, while maintaining the same low power consumption as ISCALP.

As shown in Figure 2.7, Figure 2.8, Table 2.1, Table 2.2, and Table 2.4, IFP-HLS always has better CPU time than ISCALP for both non-unity and unity aspect ratio cases except for two very small unity aspect ratio benchmarks (PAULIN and MAC). There are two contributors

Table 2.1. Numbers of Merges of Different Benchmarks

Benchmark	Unity aspect ratio			Non-unity aspect ratio		
	No. of Merges		No. of Splits	No. of Merges		No. of Splits
	ISCALP	IFP-HLS	IFP-HLS	ISCALP	IFP-HLS	IFP-HLS
CHEMICAL	593	208	24	585	190	26
DCT_DIF	981	492	1	769	508	1
DCT_IJPEG	630	380	0	850	424	11
DCT_LEE	1512	674	2	1276	691	5
DCT_WANG	974	564	10	1019	515	3
ELLIPTIC	562	172	16	533	212	11
IIR77	858	506	2	858	426	1
JACOBI_SM	1652	572	52	1755	530	26
MAC	220	31	13	200	19	14
PAULIN	87	26	6	87	25	6
PR1	839	449	10	841	448	12
PR2	1074	526	20	866	529	27
SERIAL	5200	2620	3	5200	2660	1
SMALL	11	9	0	11	13	2
WDF	1827	631	6	1588	739	9
RANDOM100	1359	511	10	1353	433	5
RANDOM200	1110	780	1	1140	780	2
RANDOM300	2810	820	0	N/A*	900	2
Average	1238.83	553.94	9.78	1113.59*	537.76*	9.11

*To solve non-unity aspect ratio Random300, ISCALP had not yet halted after 120 hours. The non-unity aspect ratio Random300 benchmark was excluded from the computation for average numbers.

to CPU time (as shown in Equation 2.3): the number of high-level synthesis moves and the resulting floorplanning operations. ISCALP employs a fast constructive slicing floorplanner based on netlist partitioning and rotation/orientation selection to obtain a floorplan optimized for wire length and area. It is faster than our simulated annealing floorplanner for small benchmarks with only a few blocks largely due to its determinism. The simulated annealing algorithm may re-visit same valid solutions multiple times before reaching the halting conditions while constructive slicing floorplanner can quickly consider all slicing structure floorplanners,

Table 2.2. CPU Times of Different Benchmarks

Benchmark	Unity aspect ratio			Non-unity aspect ratio		
	CPU Time			CPU Time		
	ISCALP (s)	IFP-HLS (s)	Speedup (×)	ISCALP (s)	IFP-HLS (s)	Speedup (×)
CHEMICAL	83.35	74.19	1.12	793.83	55.15	14.39
DCT_DIF	102.98	120.86	0.85	699.96	102.27	6.84
DCT_IJPEG	363.12	310.91	1.17	4297.13	183.36	23.44
DCT_LEE	248.07	194.41	1.28	2669.18	166.56	16.03
DCT_WANG	340.40	259.53	1.31	5678.98	229.11	24.79
ELLIPTIC	77.46	57.04	1.36	804.29	53.16	15.13
IIR77	214.89	192.87	1.11	2102.27	126.04	16.68
JACOBI_SM	1982.55	322.97	6.14	31187.15	256.60	121.54
MAC	7.64	10.28	0.74	22.12	5.64	3.92
PAULIN	1.12	2.50	0.45	3.20	2.03	1.58
PR1	162.79	138.13	1.18	2041.15	121.94	16.74
PR2	366.87	256.12	1.43	6145.86	177.36	34.65
SERIAL	1187.37	767.48	1.55	14503.25	599.59	24.19
SMALL	0.20	0.94	0.21	0.24	0.83	0.29
WDF	185.52	118.74	1.56	1092.92	118.92	9.19
RANDOM100	462.14	246.24	1.88	5951.79	204.52	29.10
RANDOM200	16438.33	3498.53	4.70	174540.95	2826.22	61.76
RANDOM300	160997.92	18786.70	8.57	N/A*	12650.64	N/A*
Average	10179.04	1408.80	2.03	14854.96*	307.61*	24.72*

*To solve non-unity aspect ratio Random300, ISCALP had not yet halted after 120 hours. The non-unity aspect ratio Random300 benchmark was excluded from the computation for average numbers.

given small enough problem sizes. In contrast, the simulated annealing floorplanner is relatively faster on large problem instances because it can focus its moves on the most promising regions of the solution space while the constructive floorplanner is left to explicitly consider an exponentially-increasing number of points in the solution space. Please note that both floorplanners run quickly on small benchmarks. We are primarily concerned with floorplanner performance on large problem instances, for which run-time is a concern. In addition, recall that ISCALP is an interconnect-aware, power-driven high-level synthesis tool. These results show

Table 2.3. CPU Times Break Down of Different Benchmarks

Benchmark	Unity aspect ratio						Non-unity aspect ratio					
	ISCALP			IFP-HLS			ISCALP			IFP-HLS		
	T_{fp} (s)	T_{total} (s)	Ratio* (%)	T_{fp} (s)	T_{total} (s)	Ratio* (%)	T_{fp} (s)	T_{total} (s)	Ratio* (%)	T_{fp} (s)	T_{total} (s)	Ratio* (%)
CHEMICAL	61.19	83.35	73.41	61.24	74.19	82.54	770.91	793.83	97.11	43.60	55.15	79.06
DCT_DIF	75.57	102.98	73.38	101.86	120.86	84.28	674.93	699.96	96.42	82.95	102.27	81.11
DCT_IJPEG	308.59	363.12	84.98	269.28	310.91	86.61	4238.01	4297.13	98.62	146.66	183.36	79.98
DCT_LEE	191.11	248.07	77.04	159.66	194.41	82.13	2616.96	2669.18	98.04	130.88	166.56	78.58
DCT_WANG	275.61	340.40	80.97	214.17	259.53	82.52	5613.21	5678.98	98.84	185.94	229.11	81.16
ELLIPTIC	49.42	77.46	63.80	47.08	57.04	82.54	780.94	804.29	97.10	41.44	53.16	77.95
IIR77	168.91	214.89	78.60	164.32	192.87	85.20	2056.25	2102.27	97.81	104.88	126.04	83.21
JACOBI_SM	1846.04	1982.55	93.11	245.88	322.97	76.13	31029.15	31187.15	99.49	188.11	256.60	73.31
MAC	4.80	7.64	62.83	9.28	10.28	90.27	19.42	22.12	87.79	4.89	5.64	86.70
PAULIN	0.54	1.12	48.21	2.11	2.50	84.40	2.50	3.20	78.13	1.66	2.03	81.77
PR1	126.84	162.79	77.92	112.32	138.13	81.31	2003.70	2041.15	98.17	96.38	121.94	79.04
PR2	299.53	366.87	81.64	213.68	256.12	83.43	6085.16	6145.86	99.01	141.40	177.36	79.72
SERIAL	974.21	1187.37	82.05	634.26	767.48	82.64	14288.12	14503.25	98.52	476.38	599.59	79.45
SMALL	0.09	0.20	45.00	0.84	0.94	89.36	0.15	0.24	62.50	0.71	0.83	85.54
WDF	126.58	185.52	68.23	90.21	118.74	75.97	1037.36	1092.92	94.92	88.49	118.92	74.41
RANDOM100	383.64	462.14	83.01	208.30	246.24	84.59	5869.24	5951.79	98.61	172.87	204.52	84.52
RANDOM200	15484.14	16438.33	94.20	2877.49	3498.53	82.25	173591.19	174540.95	99.46	2220.85	2826.22	78.58
RANDOM300	151251.05	160997.92	93.95	15798.36	18786.70	84.09	N/A**	N/A**	N/A**	10089.21	12650.64	79.75
Average	9534.88	10179.04	75.69	1178.35	1408.80	83.35	14745.72**	14854.96**	94.15**	242.83**	307.61**	80.24**

* Ratio = T_{fp}/T_{total}

**To solve non-unity aspect ratio Random300, ISCALP had not yet halted after 120 hours. The non-unity aspect ratio Random300 benchmark was excluded from the computation for average numbers.

Table 2.4. Area and Power Improvements of Different Benchmarks

Benchmark	Area Improvement (%)		Wire Power Improvement (%)		Total Power Improvement (%)	
	Unity	Non-unity	Unity	Non-unity	Unity	Non-unity
CHEMICAL	6.23	22.67	22.69	9.39	4.17	5.72
DCT_DIF	4.36	-6.91	41.51	21.86	2.37	-0.16
DCT_IJPEG	-5.09	12.35	49.59	-4.39	0.35	2.76
DCT_LEE	13.84	13.53	22.74	7.58	3.16	1.35
DCT_WANG	16.60	13.61	38.50	42.82	7.00	5.15
ELLIPTIC	9.70	9.45	22.18	28.32	2.77	0.03
IIR77	22.69	15.03	21.32	-0.37	2.42	-0.34
JACOBI_SM	22.26	24.17	14.13	9.76	-5.41	-6.51
MAC	36.13	18.84	9.65	13.72	-0.27	0.70
PAULIN	5.74	6.72	26.19	23.39	-8.72	-8.86
PR1	19.86	10.87	6.13	16.44	2.39	2.05
PR2	18.56	26.93	30.83	-7.75	2.67	1.84
SERIAL	11.16	32.04	11.74	31.57	-1.42	3.28
SMALL	22.89	13.78	36.93	48.83	1.36	1.98
WDF	8.79	11.10	9.87	-12.50	3.79	-0.91
RANDOM100	19.88	-2.41	14.67	19.94	-1.55	-4.24
RANDOM200	-1.22	12.20	19.45	-1.90	0.01	-3.01
RANDOM300	-28.55	N/A*	31.80	N/A*	-15.11	N/A*
Average	11.32	13.76*	23.89	14.51*	0.00	0.05*

*To solve non-unity aspect ratio Random300, ISCALP had not yet halted after 120 hours. The non-unity aspect ratio Random300 benchmark was excluded from the computation for average numbers.

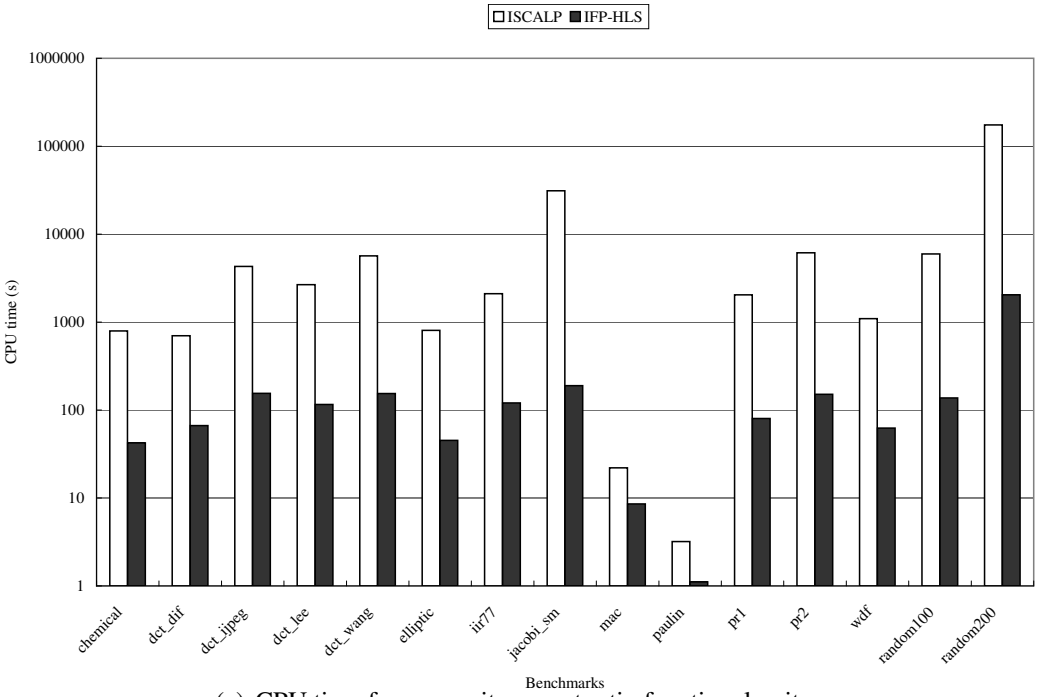
that, on average, IFP-HLS achieves better CPU time and area while maintaining good power consumption. We also analysis the time break down between high-level synthesis moves and floorplanning. As shown in Table 2.3, floorplanning used more than 75.69% of the total CPU time on average for both ISCALP and IFP-HLS; floorplanning is the most time-consuming part of the high-level synthesis design flow.

In an attempt to isolate the impact of using a constructive floorplanner from the impact of using incremental optimization, we compared the results produced by running ISCALP followed by a high-quality simulated annealing floorplanner by those produced by IFP-HLS. On average, this results in a 1.6% increase in area and 2.7% decrease in total power compared to IFP-HLS for unity aspect ratio functional units and a 0.8% increase in area and 1.3% decrease in total power consumption for non-unity aspect ratio functional units. Note that ISCALP aggressively optimizes power consumption. These results indicate that the incremental optimization algorithm within IFP-HLS permits comparable quality, using much less CPU time, compared to a non-incremental behavioral synthesis algorithm followed by an iterative improvement floorplanner.

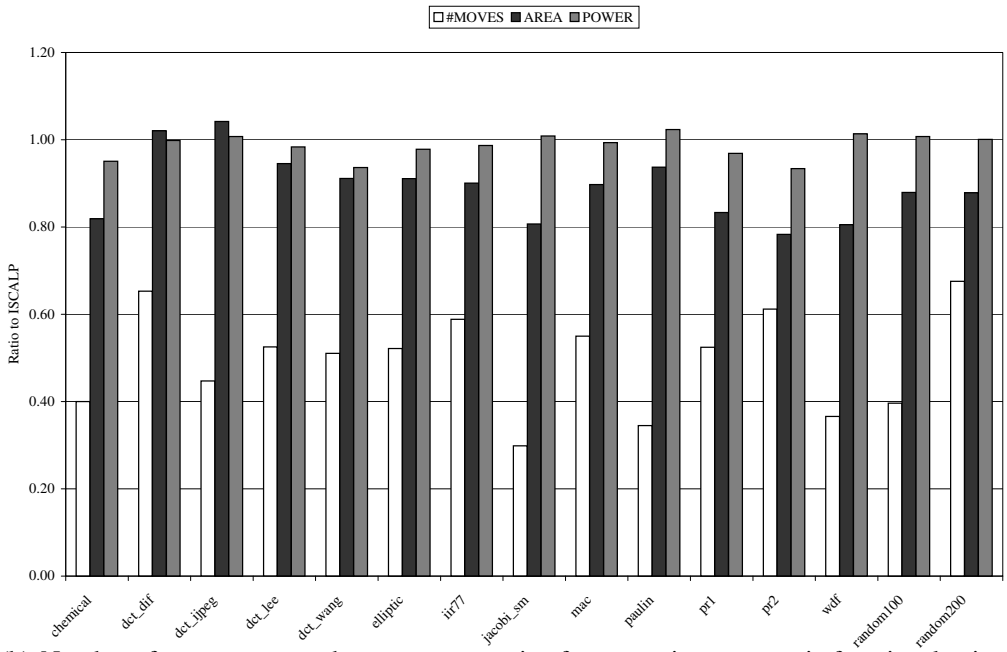
2.6. Conclusions

This chapter presented an incremental floorplanning, high-level synthesis system that integrates high-level and physical-level design algorithms to concurrently improve a design's schedule, resource binding, and floorplan. Compared with previous approaches that repeatedly call loosely coupled floorplanners, this approach has the benefit of efficiency, stability, and better quality results. As shown in Section 2.5, for non-unity aspect functional units, incremental floorplanning allowed an average CPU time speedup of 24.72× and an area improvement of

13.76%. For unity aspect ratio functional units, the CPU time speedup was 2.03× and area was improved by 11.32%. In both cases, the low power consumption of a state-of-the-art, low-power, interconnect-aware high-level synthesis algorithm was maintained. We conclude that incremental floorplanning improved the quality of synthesis results and improves performance dramatically, making synthesis from large specifications practical.

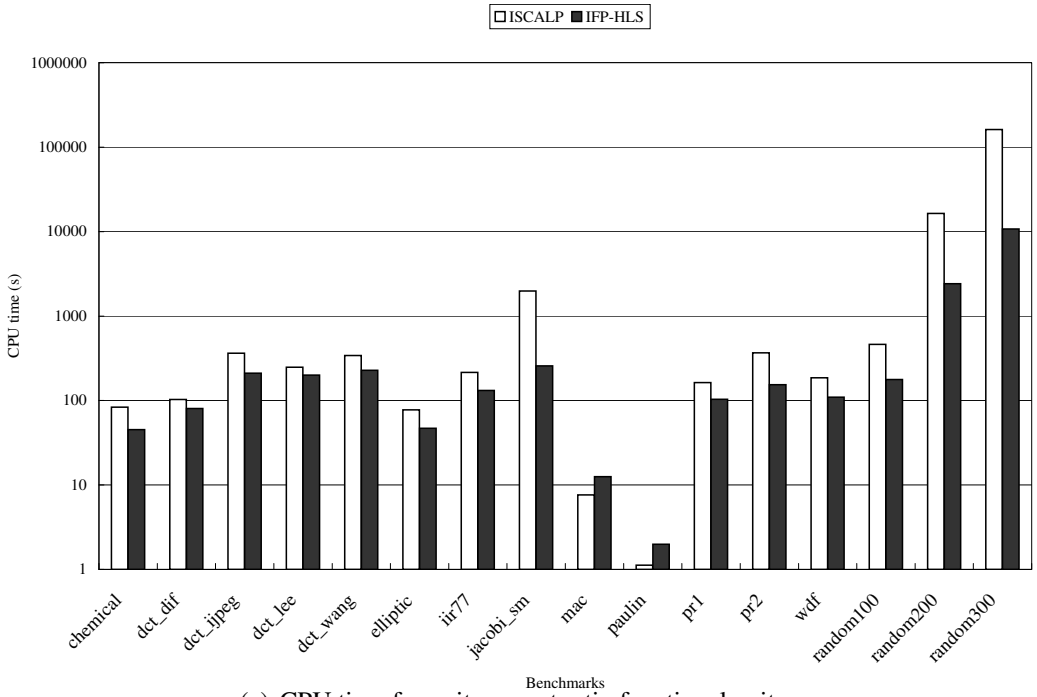


(a) CPU time for non-unity aspect ratio functional units.

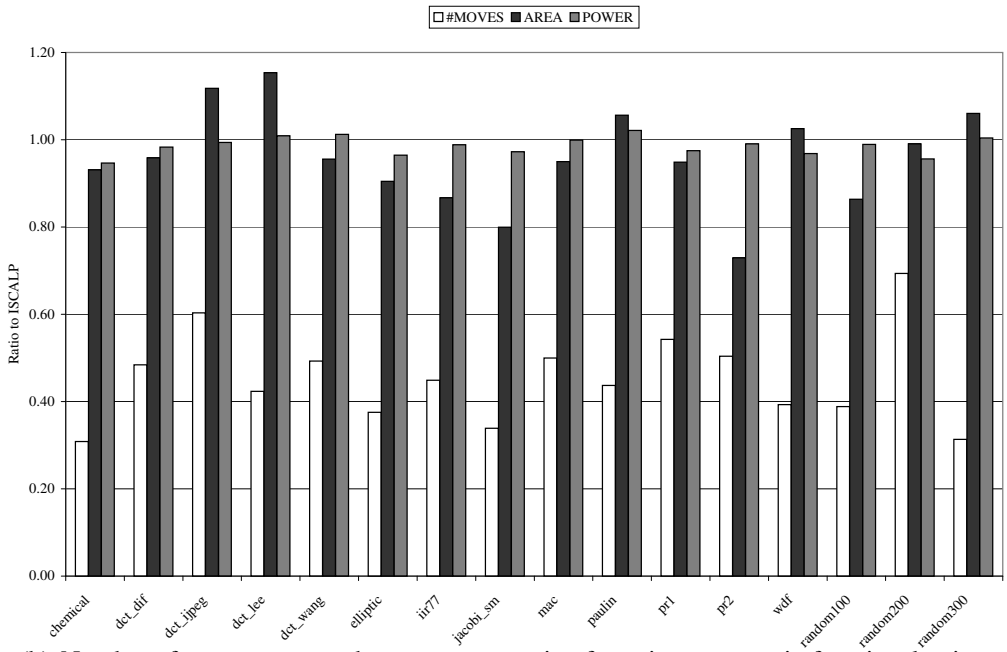


(b) Number of moves, area, and power consumption for non-unity aspect ratio functional units.

Figure 2.7. Comparison between ISCALP & IFP-HLS for non-unity aspect ratio functional units.



(a) CPU time for unity aspect ratio functional units.



(b) Number of moves, area, and power consumption for unity aspect ratio functional units.

Figure 2.8. Comparison between ISCALP & IFP-HLS for unity aspect ratio functional units.

CHAPTER 3

Unified Temperature-Aware Incremental High-Level and Physical-Level Synthesis

Thermal effects are becoming increasingly important during integrated circuit design. Thermal characteristics influence reliability, power consumption, cooling costs, and performance. It is necessary to consider thermal effects during all levels of the design process, from the architectural level to the physical level. This is challenging because design-time temperature prediction requires access to floorplans, wire models, power profile information, and a chip-package thermal model. Temperature-aware design and synthesis necessarily couple architectural-level design decisions (e.g., scheduling) with physical design (e.g., floorplanning), and modeling (e.g., wire and thermal modeling).

This chapter proposes an efficient and accurate temperature-aware high-level synthesis system that makes use of integrated high-level and physical-level design techniques. Voltage islands are automatically generated via slack distribution and voltage partitioning algorithms in order to reduce the design's power consumption and peak temperature. The proposed system was used to synthesize a number of benchmarks, yielding designs that trade off peak temperature, integrated circuit area, and power consumption. In comparison with an existing power-aware high-level synthesis algorithm, the proposed voltage control techniques reduce peak temperature by 12.5 °C on average. Under a constraint on peak temperature, integrated circuit area is reduced by 9.9% on average.

3.1. Introduction

Increasing performance requirements and system integration are dramatically increasing integrated circuit (IC) power density and, therefore, cooling costs. Thermal issues are now central to the design of ICs, including both high-end instruction processors in general-purpose computers and high-performance application-specific integrated circuits (ASICs) in portable electronic consumer devices. Peak local temperature influences the reliability, packaging costs, cooling costs, bulk, and performance of ICs; these considerations can be particularly important for portable devices.

Increasing IC power consumption raises average and peak temperatures. It is also found that temperature-dependent reliability problems account significantly for electronic failures [**6**], most of which are due to electromigration, hot carrier effects, thermal stress, and oxide temperature breakdown. Power and temperature variation can also lead to significant timing uncertainty, requiring more conservative timing margins, thereby reducing performance. Designers must frequently trade off other design metrics, such as performance, area, and cooling costs, to meet tight temperature constraints. The interaction of power and temperature constraints with other design metrics further increases system complexity. As projected by the International Technology Roadmap for Semiconductors (ITRS) [**2**], further process scaling will be bounded by power consumption and heat dissipation below 65 nm: it is critical to address the energy and thermal issues during on-chip system design to enable future technology scaling.

Thermal problems cannot be well solved at any single level of the design process. Thermal characterization requires detailed physical information, including an IC floorplan, and power profile, as well as interconnect and chip-package thermal models. Thermal optimization requires a unified high-level and physical-level design flow. At the architectural level, reducing

supply voltage can reduce IC power consumption and temperature. At the physical level, efficient floorplanner is critical to correctly implement temperature-aware techniques made by high-level design flow while maintaining other design metrics, such as performance, chip area, and cooling cost. Therefore, this requires a comprehensive high-level and physical-level infrastructure.

Incremental synthesis is a promising design technique that may be used to unify high-level synthesis and physical design. It improves the quality of results by maintaining important physical-level properties across consecutive physical design changes, many of which are triggered by architectural changes [**8**, **55**, **68**]. Moreover, it dramatically improves synthesis time by reusing and building upon high-quality, previous physical design solutions that required a huge amount of time and effort to produce.

This chapter presents an incremental temperature-aware, voltage selection, floorplanning, high-level synthesis system called TAPHS. The proposed incremental synthesis techniques rapidly learn the impact of architectural changes on floorplan-dependent characteristics (e.g., peak temperature, interconnect structure, area, and energy consumption) and concurrently optimize IC thermal profile, area, and energy consumption under performance constraints.

3.2. Related work

In this section, we survey related work in the two main research areas in which TAPHS is rooted: (1) high-level and physical-level co-synthesis and (2) temperature-aware analysis and design.

A number of researchers have considered the impact of physical details, e.g., floorplanning information, on behavioral synthesis [**41**, **42**, **43**, **45**]. Interconnect and interconnect buffers are

now first-order timing and power considerations in VLSI design [**67**]. This change has complicated both design and synthesis. It is no longer possible to accurately predict the power consumption and performance of a design without first knowing enough about its floorplan to predict the structure of its interconnect. For this reason, a number of researchers have worked on interconnect-aware behavioral synthesis algorithms [**69**, **52**, **70**, **54**]. These approaches typically use a loosely-coupled independent floorplanner for physical estimation. There are two drawbacks to this approach. First, the independent floorplanner may not be stable, i.e., a small change in the input netlist may result in a totally different floorplan. This results in a behavioral synthesis algorithm that bases its moves on cost functions without continuity. Second, even if the floorplanner is stable, creating a floorplan from scratch after each behavioral synthesis move is not efficient. The new floorplan typically only requires small changes to the previous one. Recently, incremental floorplanning and synthesis [**55**] were used to tightly couple high-level and physical synthesis that dramatically improve their combined performance and quality [**8**].

Recent studies on thermal issues focused on thermal modeling, optimization, and run-time management. Full-chip thermal modeling and analysis during synthesis were rarely considered in the past due to the formidable computational demand. Recently, a number of thermal modeling approaches, which try to model complete chip designs, have been proposed [**71**, **72**, **73**, **74**, **75**]. Architecture-level thermal modeling and management techniques were proposed to improve the thermal characteristics of microprocessors [**76**] and on-chip networks [**73**]. With increasing power densities and reducing feature sizes, temperature-related reliability problems such as electromigration and stress migration voiding are becoming increasingly important. Recent studies [**77**, **78**, **79**] have proposed numerical and analytical modeling techniques to characterize the thermal profile of on-chip interconnect layers. Recently, thermal issues have also been

considered during chip cell-level placement [**80**, **81**], three-dimensional IC floorplanning [**82**] and high-level temperature-aware resource binding and allocation [**83**]. The use of voltage islands is effective in reducing power consumption and therefore temperature. Voltage island generation has recently been incorporated with physical design [**84**, **85**, **86**, **87**].

3.3. Motivating example

In this section, we use an example IC design to demonstrate the challenges of thermal optimization in high-level synthesis. Figure 3.1 shows an IC floorplan produced by an integrated high-level synthesis and floorplanning algorithm. In this figure, the numbered rectangles are functional units, e.g., adders, multipliers, dividers, or registers. Using thermal analysis, as described in Section 3.9, the IC thermal profile is determined. The temperature of each functional unit is indicated by its brightness: brighter functional units are hotter. 85 ℃ is a typical thermal emergency threshold to ensure reliable operation. In this example, functional units temperatures higher than 85 ℃ are white. 29 of the functional units are operating at dangerously high temperatures: this chip is likely to suffer from failure caused by temperature-related reliability problems, e.g., electromigration or decreased charge carrier mobility. Note that producing the detailed chip thermal profile in Figure 3.1 requires detailed physical information, i.e., a floorplan, a power profile, and a chip-package thermal model. Therefore, stand-alone high-level synthesis algorithms have no means of detecting, let alone correcting, thermal problems.

High-level synthesis provides numerous thermal optimization opportunities. Reducing supply voltage reduces power consumption, hence temperature, but may also impair performance. Recent work on voltage islands has proposed operating different regions of an IC at different voltages. Figure 3.2 illustrates the floorplan of an IC using voltage islands. In this design,

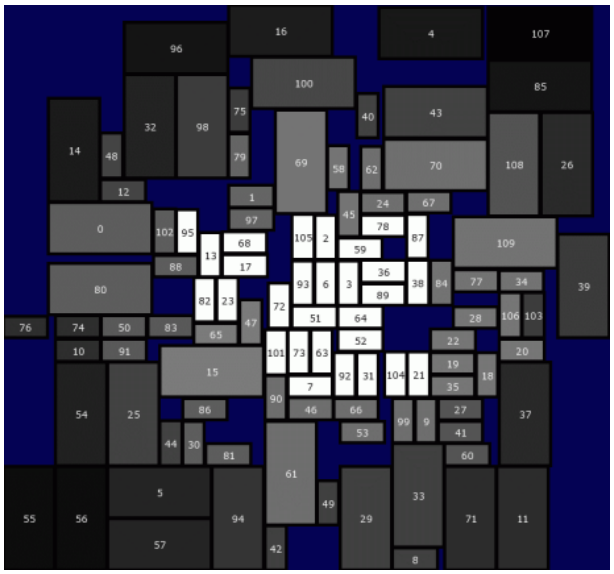


Figure 3.1. Post-synthesis thermal profile without voltage islands.

functional units are assigned to contiguous voltage islands with different supply voltages. The brightnesses of the thick functional unit boundaries indicate their voltages. In this example, three voltage islands are used. As in Figure 3.1, functional units violating the 85 ℃ temperature constraint are white.

A comparison of Figures 3.1 and 3.2 indicates that voltage islands can dramatically improve thermal conditions. The number of functional units with temperatures above the temperature constraint decreased from 29 to 19. However, as shown in Figure 3.2, localized hot spots still exist. The remaining hot spots are primarily the result of local peaks in power density. Therefore, thermal analysis algorithms are invoked to guide optimization moves in the high-level design.

Our study suggests a set of high-level and physical-level thermal optimization techniques, including multiple operating voltages and appropriate scheduling. Many of the techniques to

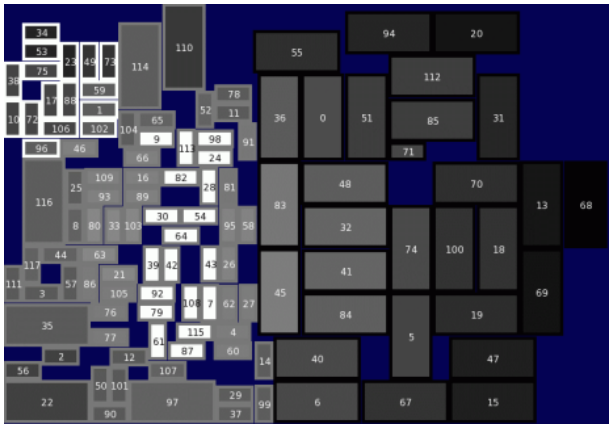


Figure 3.2. Post-synthesis thermal profile with voltage islands.

optimize IC thermal properties also impact other design metrics such as area and power consumption. We have considered the side effects of a number of techniques, proposing those that allow improvements to thermal properties while maintaining good area, performance, and power consumption.

Using voltage islands has a significant impact on chip area and performance as well as increasing the complexity of floorplanning. Voltage islands require the addition of voltage converters and delivery circuits, as well as on-chip level shifters to support communication among functional units in different voltage islands. Moreover, reduced supply voltage requires a longer clock period to compensate for reduced switching speeds. In order to use voltage islands, a synthesis algorithm must wisely choose the island for each functional unit, appropriately allocate timing slack to allow scheduling in the high-level design. In the physical-level design, high-quality incremental floorplans is necessary to form voltage islands based on voltage assignment from high-level decision as well as maintaining other design metrics such as area, total wire length. This tightly couples the architectural and physical levels of design.

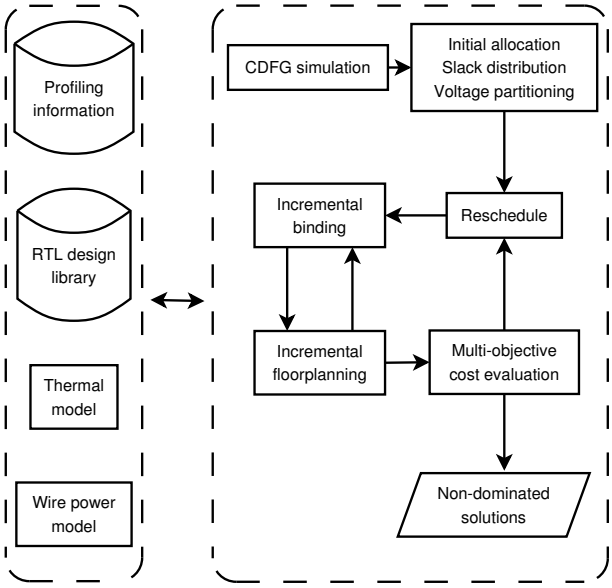


Figure 3.3. Incremental high-level synthesis algorithm

Facing these design challenges, a high-quality temperature-aware synthesis system must incorporate thermal optimization techniques into a unified high-level and physical level design flow, as well as striking wise tradeoffs among conflicting design goals.

3.4. Overview of TAPHS

In this section, we give an overview of TAPHS: our incremental temperature-aware physical and high-level synthesis system. TAPHS considers the thermal impact of both logic and interconnect power dissipation. It automatically plans voltage islands and schedules operations to reduce IC power consumption and peak temperature.

Figure 3.3 illustrates the main algorithms used in TAPHS. First, the control and data flow graph is simulated with typical input traces in order to determine the power consumption of each operation and data transfer edge. The profile information, an RTL design library, floorplanner, and thermal model are used to evaluate the IC temperature profile, power, area, and

performance. Slack distribution, voltage clustering, and voltage island aware floorplanning are used to generate voltage islands for use in the initial solution: a fully parallel implementation. There are two loops within the high-level synthesis algorithm. In the outer loop, the clock period of the design is iteratively changed from the minimum to maximum potentially feasible value. Incremental rescheduling, resource sharing, resource splitting (i.e., the opposite of resource sharing), and slack distribution are used to generate valid solutions. In the inner loop, back-tracking iterative improvement is used to optimize the RTL architecture, considering multiple objectives, e.g., peak temperature, area, or power consumption. A *dominated* solution is inferior to some other previously encountered solution in all costs. Non-dominated solutions are preserved in a solution cache, from which the designer may choose based upon the desired, and available, trade-offs among costs.

A high-quality incremental floorplanner was developed [**88**] and incorporated into TAPHS. Each time the high-level synthesis algorithm needs temperature and physical information to guide its moves, it extracts that information from the current, incrementally generated, floorplan. In addition, costs derived from the floorplan are also used to guide high-level synthesis moves. By using incremental floorplanning, closer interaction between high-level synthesis and physical design is possible, i.e., the high-level synthesis algorithm may determine the impact of potential changes to binding upon physical attributes such as maximum IC temperature, area, and interconnect energy consumption.

3.5. Slack distribution

In order to allow voltage scaling, it is necessary to appropriately distribute scheduling slack among different operations. An operation's *slack* is the difference between its latest start time

and earliest start time. If operation start times are determined with an as-soon-as-possible schedule (ASAP), the executions of most operations will be immediately followed by other operations. As a result, determining whether it is possible, and desirable, to assign an operation to a lower-voltage functional unit without violating timing constraints is not possible based on ASAP operation start times. TAPHS redistributes slack among operations in order to support a reduced energy assignment of functional units to voltage islands. Note that voltage assignment (described in Section 3.6) may not arrive at an energy-optimal solution, as it is necessary to constrain the off-chip overhead that would result from numerous power regulators. As shown in Figure 3.3, slack distribution occurs before voltage partitioning.

Assume that control data flow graphs have been partitioned into same-slack paths, as described later in this subsection. Given a single path composed of sequential operations, the slack distribution problem is equivalent to deciding the execution times of each operation such that energy consumption is minimized under a hard constraint on path execution time. We shall use the following variables and constants:

- D is the bound on path execution time;
- p is the set of all operations on the path;
- v is the voltage of an operation's functional unit;
- V_t is the threshold voltage constant;
- K is an execution time constant;
- E is the total path energy consumption;
- e is the energy required for an operation;
- C is the switched capacitance constant of an operation's functional unit; and
- α is the alpha power law constant [**89**].

$$d = \frac{Kv}{(v - V_t)^{\alpha}} \tag{3.1}$$

However, V_t is small and a very low value of v will generally imply an unacceptable path delay that will be prevented by the constraint on line 3.7. Therefore, we may assume V_t is small, thus

$$d \simeq \frac{Kv}{v^{\alpha}} \tag{3.2}$$

$$v = \left(\frac{d}{K}\right)^{\frac{1}{1-\alpha}} \qquad \text{by (3.2)} \tag{3.3}$$

$$e = Cv^2 \tag{3.4}$$

$$e = c\left(\frac{d}{K}\right)^{\frac{2}{1-\alpha}} \qquad \text{by (3.2) and (3.4)} \tag{3.5}$$

$$E = \sum_{i \in p} C_i \left(\frac{d_i}{K_i}\right)^{\frac{2}{1-\alpha}} \tag{3.6}$$

$$\min_{\substack{\forall i \in p \\ v_i}} \quad \sum_{i \in p} C_i \left(\frac{d_i}{K_i}\right)^{\frac{2}{1-\alpha}} \text{ subject to the constraint } D \geq \sum_{i \in p} d_i \tag{3.7}$$

Note that a decrease in v implies an decrease e, which implies an increase in d. Therefore, for minimal E,

$$D = \sum_{i \in p} d_i \tag{3.8}$$

Consider the delay and energy trade-off for an arbitrary pair of operations:

$$d_{12} = d_1 + d_2 \tag{3.9}$$

$$e_{12} = e_1 + e_2 \tag{3.10}$$

$$e_{12} = \frac{C_1}{K_1^{\frac{2}{1-\alpha}}}(d_1)^{\frac{2}{1-\alpha}} + \frac{C_2}{K_2^{\frac{2}{1-\alpha}}}(d_{12} - d_1)^{\frac{2}{1-\alpha}} \tag{3.11}$$

Take the derivative of e_{12} with respect to d_1, set to zero, and solve to find d_2/d_1 for minimal E.

$$\frac{d_2}{d_1} = \left(\frac{\frac{C_1}{K_1^{\frac{2}{1-\alpha}}}}{\frac{C_2}{K_2^{\frac{2}{1-\alpha}}}} \right)^{\frac{1-\alpha}{1+\alpha}} \tag{3.12}$$

This optimal delay ratio for two operations may be used to compute the optimal delay ratio for an arbitrary pair of operations. These ratios can be scaled by a dynamically computed value, N, to ensure that the constraint on line 3.7 is honored.

$$N = \sum_{i \in p} \frac{d_i}{d_1} \tag{3.13}$$

$$\forall_{i \in p}\ d_i = \left(\frac{\frac{C_1}{K_1^{\frac{2}{1-\alpha}}}}{\frac{C_i}{K_i^{\frac{2}{1-\alpha}}}} \right)^{\frac{1-\alpha}{1+\alpha}} \frac{D}{N} \qquad \text{by (3.11)} \tag{3.14}$$

$$\forall_{i \in p}\ d_i = \sqrt[3]{\frac{C_i K_i^2}{C_1 K_1^2}} \cdot \frac{D}{N} \qquad \text{by fixing } \alpha = 2 \tag{3.15}$$

Equations 3.11 and 3.15 yield the optimal time, d_i, to dedicate to each operation. By granting slack to each operation in the path such that its time is proportional to its time share, we

allow the voltage island generation algorithm the opportunity to assign functional units to voltage islands such that energy consumption may be minimized under a hard constraint on path execution time (please see Section 3.6).

Thus far, we have discussed individual operation paths. However, it is necessary for TAPHS to determine slack distributions along numerous paths in arbitrary directed acyclic graphs of operations. Assigning time shares eventually has the effect of (temporarily) fixing operation start times. These start times may influence the earliest start times and latest finish times of operations on other paths; in order to avoid deadline violations, slack distribution is conducted on operation paths in order of increasing path slack. In order to generate paths, a modified depth-first search is conducted on a graph in which each vertex is an operation labeled with its slack and each edge is a data dependency. Vertex children are visited in the order of increasing slack, thereby guaranteeing that vertices on multiple paths will be included in minimal-slack paths.

Algorithm 2 Slack distribution procedure

1: Compute all operation slacks
2: Group operations into same-slack paths, P
3: Sort paths P in order of increasing slack
4: **for all** $p \in P$ **do**
5: **while** slack remains on p **do**
6: $\forall_{i \in p}\ t_i$ is the time assigned to operation i
7: Operation $i = \underset{j}{\arg\min} \sqrt[3]{\frac{C_i {K_i}^2}{C_1 {K_1}^2}} \cdot \frac{D}{N} - t_j$ by Equation 3.15
8: Assign one additional clock cycle to operation i
9: **end while**
10: Recompute all operation slacks
11: **end for**

As shown in Algorithm 2, starting from the minimal-slack path, TAPHS incrementally assigns extra clock cycles to operations. At each step, it locates the operation, j, for which the

current allocated time, t_j, differs most from d_j (Step 8) and assigns it an additional clock cycle (Step 9). It is unlikely that this will result in deadline violations on other paths because slack distribution is carried out on paths in order increasing slack. Therefore, slack distribution on a given path is prevented from delaying any node so much that slack becomes negative on other paths on which the node lies. After slack sharing is done for a given path, the slacks of all nodes are recomputed and slack distribution proceeds for the next path. The proposed slack distribution algorithm is optimal for continuous slack values. Discretization may introduce sub-optimality. Recently, Ghiasi et al. [**90**] proposed a min-cost flow algorithm that optimally solves problem with linear cost functions. However, it is not applicable for this problem because the cost function is nonlinear.

3.6. Voltage partitioning

TAPHS uses on-chip voltage islands to optimize IC thermal profiles and energy consumption. On-chip voltage islands are generated in two stages. *Voltage partitioning* classifies on-chip functional units into different voltage levels to maximize overall power and energy savings hence potential chip temperature reduction. *Voltage island generation* is then conducted via incremental floorplanning to produce and optimize on-chip voltage islands.

In this work, we propose an efficient voltage partitioning algorithm. It conducts optimal voltage allocation and assignment to minimize power consumption subject to timing constraints.

Motivating example We next present an example to illustrate the voltage partitioning issue. Consider a circuit design with five functional units as shown in Fig. 3.4. For each functional unit, FU_i, the minimal allowed supply voltage, $V_{FU_i}^{min}$, is uniquely determined by the ratio of its time slack to its propagation delay under the initial (maximum) supply voltage. Then, in a

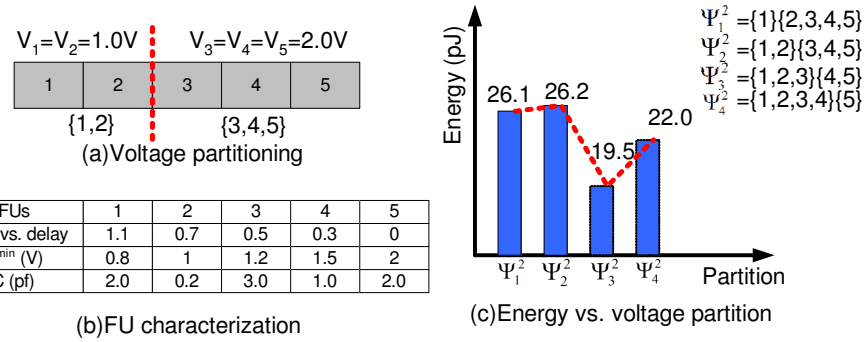


Figure 3.4. Voltage partitioning example.

voltage partition Ψ_i^S with S clusters, to satisfy the deadline constraints of functional units, for each cluster, $\psi_j = \{FU_{j1}, \ldots, FU_{jn}\}$, $\psi_j \in \Psi_i^S$, the supply voltage, V_{ψ_j}, is greater than or equal to $\max\{V_{FU_{jt}}^{\min}\}_{t=1,\ldots,n}$, i.e., the minimal supply voltage of the functional unit with the lowest slack-delay ratio inside this cluster. Consider the voltage partitioning shown in Fig. 3.4(a). This partitioning contains two voltage clusters, $\psi_1 = \{FU_1, FU_2\}$ and $\psi_2 = \{FU_3, FU_4, FU_5\}$. The supply voltage of ψ_1, $V_{\psi_1} = 1\,\mathrm{V}$, which is the minimal allowed supply voltage of FU_2. The supply voltage of ψ_2, $V_{\psi_2} = 2\,\mathrm{V}$, which is the minimal allowed supply voltage of FU_5.

Fig. 3.4(c) shows the energy consumptions of different voltage partitions, which are derived using a linear scan along the functional unit list. This list is sorted in order of increasing slack-delay ratio (or minimal allowed supply voltage) of functional units. This figure shows that, using linear scan, the energy curve is not monotonic, implying that an algorithm with $O(N)$ time complexity is required to find a single voltage partitioning cut with minimal energy consumption.

Next, let us define the optimal voltage partitioning problem.

Problem Definition *Given N functional units, $\{FU_i\}$, and an input K, we need to find an optimal voltage partition, Ψ_{opt}^K, containing K voltage clusters, $\{\psi_{opt\,j}\}_{j=1,\ldots,K}$, such that the*

total energy consumption, i.e., $E(\Psi^K_{opt}) \leq E(\Psi^K_i), \forall \Psi^K_i$*, in which* $E(\Psi^K_{opt}) = \sum_{l=1}^{N} C_l \times V^2_{\psi_{opt\,j}}$*.* C_l *is the capacitance of* $FU_l, FU_l \in \psi_{opt\,j}, j = 1, \ldots, K$*.*

For each functional unit, FU_i, to satisfy its deadline constraint, its minimal allowed supply voltage, $V^{min}_{FU_i}$, is uniquely determined by the ratio of its slack time to its propagation delay under the initial (maximum) supply voltage. Then, for each cluster, $\psi_j = \{FU_{j1}, \ldots, FU_{jn}\}$, $\psi_j \in \Psi^K_i$, its supply voltage, $V_{\psi_j} \geq \max\{V^{\min}_{FU_{jt}}\}_{t=1,\ldots,n}$, i.e., the minimal supply voltage of the functional unit with the lowest slack to delay ratio inside this cluster.

An optimal voltage partitioning is derived using the following approach. Functional units are first sorted based on their slack to propagation delay ratios. Then, linear scans along the sorted functional unit list determine the optimal partitioning. Notice that the energy saving curve is not monotonic, suggesting that an algorithm with $O(N)$ time complexity is required to find an energy-optimal voltage partitioning. For K partitions, the time complexity of this algorithm is $O\left(N^K\right)$.

An optimal voltage partitioning algorithm of $O\left(N^2\right)$ complexity

We introduce an optimal voltage partitioning algorithm of $O\left(N^2\right)$ time complexity. Its pseudo-code is shown in Algorithm 3, which is described in a recursive form. *Partition*() has five input/output parameters. $*FU\_list$ points to the sorted functional unit list. *Start* and *End* designate the sub-list, the portion of the original list that need to be partitioned. Initially, $Start = 0$ and $End = N$ denote voltage partitioning targets on the whole sorted list. K defines the targeted number of partition cuts. $OptTable$ stores intermediate optimal partitions of sub-lists.

Partition() is invoked recursively when $K > 1$ (line 1–4). For each sub-partitioning (K cuts) on a sub-list (from *Start* to *End*), the optimal solution is derived using linear scan to examine the Kth cut from *Start* to *End*, which is combined with the optimal solution of the sub-partitioning

($K-1$ cuts) on its sub-list (from i to End). When $K=1$, the algorithm uses a linear scan to find the optimal cut in the targeted sub-list (line 12).

Algorithm 3 $Partition(*FU\_list, Start, End, K, *OptTable)$

1: **if** $K > 1$ **then**
2: $\quad C \leftarrow 0$
3: $\quad$ **for** $(i \leftarrow Start; i \leq End; i++)$ **do**
4: $\quad\quad Partition(*FU\_list, i, End, K--, *OptTable)$
5: $\quad\quad E^K_{K^{th}=i} \leftarrow C \times (V^{min}_{FU_{i-1}})^2 + OptTable[K-1][i]$
6: $\quad\quad C += C_{FU_i}$
7: $\quad$ **end for**
8: $\quad E^K_{opt}(Start, End) \leftarrow min\{E^K_{K^{th}=i}\}_{i \leftarrow Start,\ldots,End}$
9: $\quad cut^K_{opt}(Start, End) \leftarrow i \; if \; E^K_{K^{th} \leftarrow i} = E^K_{opt}(Start, End)$
10: $\quad OptTable[K][Start] \leftarrow pair(E^K_{opt}(Start, End), cut^K_{opt}(i, End))$
11: **else**
12: $\quad Linear\_Scan(*FU\_list, End, \&E^1_{opt}(Start, End),$
$\quad \&cut^1_{opt}(Start, End))$
13: $\quad OptTable[1][Start] \leftarrow pair(E^1_{opt}(Start, End),$
$\quad cut^1_{opt}(Start, End))$
14: **end if**

Lemma 1 *In an optimal partition Ψ^K_{opt} with K voltage clusters, $\{\psi_{opt,1}, \ldots, \psi_{opt,K}\}$, ordered by increasing voltage, $V_{\psi_{opt,1}} < V_{\psi_{opt,2}} < \cdots < V_{\psi_{opt,K}}$, then the minimal allowed voltage of any functional unit FU_j in the optimal i_{th} voltage cluster is greater than the voltage level of adjacent lower voltage cluster, i.e., $V_{\psi_{opt,i-1}} < V^{min}_{FU_j} \leq V_{\psi_{opt,i}}, \forall FU_j \in \psi_{opt,i}$, where $V^{min}_{FU_j}$ is the minimum allowed voltage supply.*

Proof: Assume there exists a $FU_j \in \psi_{opt,i}$ such that $V_{\psi_{opt,i-1}} \geq V^{min}_{FU_j}$. If $V^{min}_{FU_j} = V_{\psi_{opt,i-1}}$, this contradicts the claim $V_{\psi_{opt,i}} > V_{\psi_{opt,i-1}}$. If $V^{min}_{FU_j} < V_{\psi_{opt,i-1}}$, then we can simply move FU_j to partition $\psi_{opt,i-1}$, which results in a lower energy consumption partition; note that $V_{\psi_{opt,i}} \geq V_{\psi_{opt,i-1}}$. This contradicts the claim that partition Ψ^K_{opt} is optimal, completing our proof.

Lemma 1 implies that the optimal partitioning can be found by partitioning the sorted functional unit list. This lemma guarantees the optimality of the algorithm: it uses combinational linear scans to explore all the possible partitioning combinations of the sorted list, including the optimal solution. The time complexity of using a combinational linear scan to find the optimal K partitions on a sorted list with N functional units in $O\left(N^K\right)$. To improve computation efficiency, we use a data structure, called *OptTable*, to store optimal sub-partitions. The time complexity of K cuts partitioning results from a linear scan of the Kth cut multiplied by the time complexity of finding the optimal $K-1$ partitions. This only requires a linear search in *OptTable* table (line 5) with complexity $O(N)$. In total, there are K recursive layers. Since K is much smaller than N, the overall time complexity of this optimal voltage partitioning algorithm is $O\left(N^2\right)$.

Preprocessor algorithm with $O(N)$ complexity

In TAPHS, we also implement an efficient preprocessing algorithm with $O(N)$ complexity to further improve the efficiency of voltage partitioning. The preprocessing algorithm contains two steps, *right-shift* and *left-shift*. We explain the *right-shift* algorithm here, the *left-shift* algorithm is complementary.

The *right-shift* partitioning algorithm is incremental. For voltage partitioning with K cuts, this algorithm starts from the optimal solution of $K-1$ cuts. Considering the following initial partitioning: $cut_1^K = 0, cut_2^K = cut_{opt,1}^{K-1}, \ldots, cut_{K-1}^K = cut_{opt,K-2}^{K-1}, cut_K^K = cut_{opt,K-1}^{K-1}$. The preprocessor right shifts each cut, i.e., $cut_i^K = cut_i^K ++, i = 1, \ldots, K$, to find optimal voltage partitioning solution with K cuts, following the order from 1 to K. Each cut is right shifted if and only if this results in a decrease in overall energy consumption. The algorithm stops when further shifts can not reduce overall energy any more. The correctness of this algorithm is guaranteed

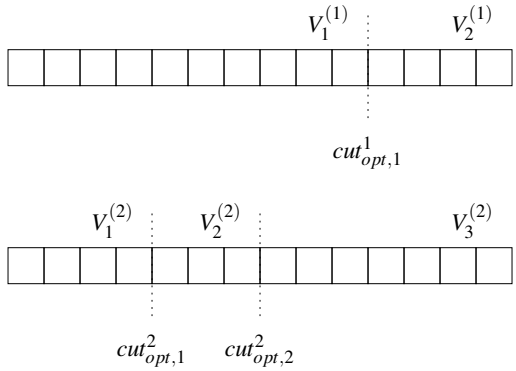


Figure 3.5. Lemma 2 with $K = 2$.

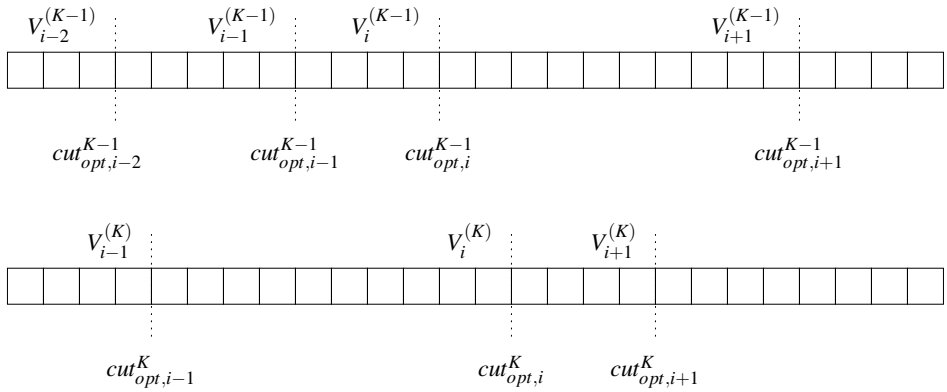


Figure 3.6. Property 1 of general K cuts optimal solution for Lemma 2.

by Lemma 2. In addition, Lemma 2 implies that this algorithm requires at most N steps to reach optimality, yielding time complexity in $O(N)$. However, this is a greedy algorithm: it may become trapped at a local minimal. Therefore, it cannot guarantee global optimality.

Lemma 2 In a sorted functional unit list with increasing minimum allowed voltage supply, if the optimal $K-1$ partitions are $\{cut_{opt,1}^{K-1}, \ldots, cut_{opt,K-1}^{K-1}\}$, and the optimal K partitions are $\{cut_{opt,1}^{K}, \ldots, cut_{opt,K}^{K}\}$, then $cut_{opt,i}^{K}$ must be in the range between $cut_{opt,i-1}^{K-1}$ and $cut_{opt,i}^{K-1}$, i.e., $cut_{opt,i-1}^{K-1} \leq cut_{opt,i}^{K} \leq cut_{opt,i}^{K-1}$.

Proof: First, we will give the proof for K=2, then extend to general case with the proof of two properties which can guarantee the correctness of lemma 2.

Given $cut^1_{opt,1}$ is the optimal partition for one partition cut. The voltage of two clusters are $V_1^{(1)}$ and $V_2^{(1)}$. Shown in Fig. 3.5, assume for K=2, both two cuts ($cut^2_{opt,1}$ and $cut^2_{opt,2}$) of the optimal partitions are on the left of $cut^1_{opt,1}$. The voltage of three clusters are $V_1^{(2)}$, $V_2^{(2)}$, and $V_3^{(2)}$. Let's move the $cut^2_{opt,2}$ to the same place as optimal one cut partition, $cut^1_{opt,1}$. Therefore, the change on energy can be as follows:

$$E^2_{inc} = \sum_{FU_i \in A} C_i((V_2^{(2)})^2 - (V_3^{(2)})^2) \tag{3.16}$$

$$E^2_{dec} = \sum_{FU_i \in B} C_i((V_3^{(2)})^2 - (V_2^{(1)})^2) \tag{3.17}$$

Here set A is the functional units between $cut^2_{opt,2}$ and $cut^1_{opt,1}$. Set B is the functional units between $cut^1_{opt,1}$ and the end of functional list. We also can move $cut^1_{opt,1}$ to the same position as $cut^2_{opt,2}$ and get energy changes as follows:

$$E^1_{inc} = \sum_{FU_i \in B} C_i((V_3^{(2)})^2 - (V_2^{(1)})^2) \tag{3.18}$$

$$E^1_{dec} = \sum_{FU_i \in A} C_i((V_1^{(1)})^2 - (V_3^{(2)})^2) \tag{3.19}$$

Since $cut^1_{opt,1}$ is the optimal cut for one cut partition. Hence, $E^1_{inc} > E^1_{dec}$. We also can find that $E^2_{dec} = E^1_{inc}$, and $E^2_{dec} < E^1_{inc}$. Therefore, we can get $E^2_{inc} < E^2_{dec}$. This means that there is an energy saving by moving $cut^2_{opt,1}$ to the same place as optimal one cut partition. This contradicts the fact that the current partition is optimal. In the same way, we can proof that both two cuts of the optimal partitions are on the right of $cut^1_{opt,1}$ is impossible.

Now, let's extend above proof to general case. We will give following two properties for an optimal K partitioning solution: (1) there must has at least a cut between position $cut^{K-1}_{opt,i-1}$ and $cut^{K-1}_{opt,i}$; (2) if there are two cuts between position $cut^{K-1}_{opt,i-1}$ and $cut^{K-1}_{opt,i}$, then there must be a cut between $cut^{K-1}_{opt,i}$ and $cut^{K-1}_{opt,i+1}$, and a cut between $cut^{K-1}_{opt,i-2}$ and $cut^{K-1}_{opt,i-1}$. Based on these two properties, we can find the optimal solution follows the rule $cut^{K-1}_{opt,i} \leq cut^{K}_{opt,i} \leq cut^{K-1}_{opt,i-1}$ because if there exists a solution satisfied property 2, then it can not meet the requirement for property 1. Therefore, the only optimal solution is that there exists one and only one cut between the position $cut^{K-1}_{opt,i}$ and $cut^{K-1}_{opt,i-1}$.

Property 1: Shown in Fig. 3.6, given optimal solution of voltage partitioning with $K-1$ cuts ($\{cut^{K-1}_{opt,1}, \ldots, cut^{K-1}_{opt,K-1}\}$) and optimal solution of voltage partitioning with K cuts ($\{cut^{K}_{opt,1}, \ldots, cut^{K}_{opt,K}\}$), assume that $cut^{K}_{opt,i}$ is on the right of $cut^{K-1}_{opt,i}$, i.e. no cut between position $cut^{K-1}_{opt,i-1}$ and $cut^{K-1}_{opt,i}$.

First, by moving $cut^{K}_{opt,i}$ into the same position as $cut^{K-1}_{opt,i}$, we can have following energy chances:

$$E^{K}_{inc} = \sum_{FU_i \in A} C_i((V^{(K)}_{i+1})^2 - (V^{(K)}_{i})^2) \tag{3.20}$$

$$E^{K}_{dec} = \sum_{FU_i \in B} C_i((V^{(K)}_{i})^2 - (V^{(K-1)}_{i})^2) \tag{3.21}$$

Here set A is the functional units between $cut^{K}_{opt,i}$ and $cut^{K-1}_{opt,i}$. Set B is the functional units between $cut^{K-1}_{opt,i}$ and $cut^{K}_{opt,i-1}$.

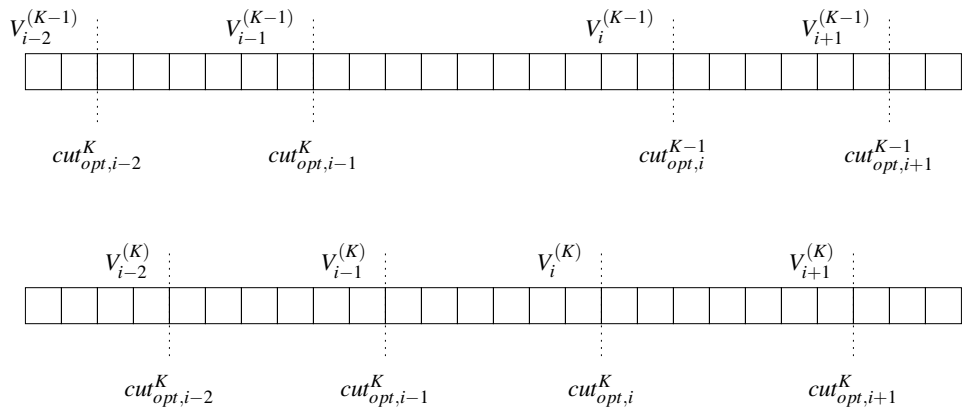


Figure 3.7. Property 2 of general K cuts optimal solution for Lemma 2.

Second, by moving $cut_{opt,i}^{K-1}$ into the same position as $cut_{opt,i}^{K}$, we can have following energy chances:

$$E_{inc}^{K-1} = \sum_{FU_i \in C} C_i((V_i^{(K)})^2 - (V_i^{(K-1)})^2) \tag{3.22}$$

$$E_{dec}^{K-1} = \sum_{FU_i \in A} C_i((V_{i+1}^{(K-1)})^2 - (V_i^{(K)})^2) \tag{3.23}$$

Here, set C is the functional units between $cut_{opt,i}^{K-1}$ and $cut_{opt,i-1}^{K-1}$. Since $cut_{opt,i}^{K-1}$ is the optimal cut for $K-1$ partitions. Hence, $E_{inc}^{K-1} > E_{dec}^{K-1}$. From above equation, we can see that $E_{dec}^{K-1} > E_{inc}^{K}$ since set C is a subset of set B. We also can get $E_{dec}^{K-1} > E_{inc}^{K}$ because $V_{i+1}^{(K-1)} > V_{i+1}^{(K)}$. Therefore, $E_{dec}^{K} > E_{inc}^{K}$. This contradicts the fact that the current K partitions is optimal. In the same way, we can proof that $cut_{opt,i}^{K}$ is on the left of $cut_{opt,i-1}^{K-1}$. Therefore, $cut_{opt,i}^{K}$ must be in the scope between position $cut_{opt,i}^{K-1}$ and $cut_{opt,i-1}^{K-1}$.

Property 2: Shown in Fig. 3.7, given optimal solution of voltage partitioning with $K-1$ cuts ($\{cut_{opt,K-1}^{K-1}, \ldots, cut_{opt,1}^{K-1}\}$) and optimal solution of voltage partitioning with K cuts ($\{cut_{opt,K}^{K}, \ldots, cut_{opt,1}^{K-1}\}$), assume that $cut_{opt,i}^{K}$ and $cut_{opt,i-1}^{K}$ are in the scope between position $cut_{opt,i}^{K-1}$ and $cut_{opt,i-1}^{K-1}$ and there is no cut between $cut_{opt,i+1}^{K-1}$ and $cut_{opt,i}^{K-1}$.

First, by moving $cut^{K}_{opt,i}$ into the same position as $cut^{K-1}_{opt,i}$, we can have following energy chances:

$$E^{K}_{inc} = \sum_{FU_i \in B} C_i((V_i^{(K-1)})^2 - (V_i^{(K)})^2) \tag{3.24}$$

$$E^{K}_{dec} = \sum_{FU_i \in A} C_i((V_{i+1}^{(K)})^2 - (V_i^{(K-1)})^2) \tag{3.25}$$

Here set A is the functional units between $cut^{K-1}_{opt,i}$ and $cut^{K}_{opt,i}$. Set B is the functional units between $cut^{K}_{opt,i}$ and $cut^{K}_{opt,i-1}$.

Second, by moving $cut^{K-1}_{opt,i}$ into the same position as $cut^{K}_{opt,i}$, we can have following energy chances:

$$E^{K-1}_{inc} = \sum_{FU_i \in A} C_i((V_{i+1}^{(K-1)})^2 - (V_i^{(K-1)})^2) \tag{3.26}$$

$$E^{K-1}_{dec} = \sum_{FU_i \in C} C_i((V_i^{(K-1)})^2 - (V_i^{(K)})^2) \tag{3.27}$$

Here, set C is the functional units between $cut^{K}_{opt,i}$ and $cut^{K-1}_{opt,i-1}$. Since $cut^{K-1}_{opt,i}$ is the optimal cut for $K-1$ partitions. Hence, $E^{K-1}_{inc} > E^{K-1}_{dec}$. From above equation, we can see that $E^{K-1}_{dec} > E^{K}_{inc}$ since set C is a subset of set B. We also can get $E^{K-1}_{dec} > E^{K}_{inc}$ because $V^{(K)}_{i+1} > V^{(K-1)}_{i+1}$. Therefore, $E^{K}_{dec} > E^{K}_{inc}$. This contradicts the fact that current K partitions is optimal. Therefore, there must be a cut between position $cut^{K-1}_{opt,i+1}$ and $cut^{K-1}_{opt,i}$. In the same way, we can proof there must be a cut between position $cut^{K-1}_{opt,i-1}$ and $cut^{K-1}_{opt,i-2}$.

Similarly, we use *left-shift* to further reduce the search space. For *left-shift*, the initial partitions start from $cut^{K}_{K} = cut^{K-1}_{opt,K-1}, \ldots, cut^{K}_{2} = cut^{K-1}_{opt,1}, cut^{K}_{1} = N$. Then, this algorithm shifts each cut to the left, i.e., $cut^{K}_{i} = cut^{K}_{i} - -$, $i = 1, \ldots, K$, to explore optimal K partitions. Overall,

for a voltage partitioning with K cuts, using these two preprocessing steps, the optimal algorithm only needs to search the following space to find optimal partitions
$cut^{K}_{opt,K} \in \{cut^{K}_{right\text{-}shift,K}, cut^{K}_{left\text{-}shift,K}\}, \ldots, cut^{K}_{opt,1} \in \{cut^{K}_{right\text{-}shift,1}, cut^{K}_{left\text{-}shift,1}\}$.

3.7. Floorplanning with voltage islands

In order to support temperature-aware, incremental, unified high-level and physical-level optimization, it was necessary to incorporate a high-quality, incremental floorplanner within TAPHS. New algorithms were developed and incorporated into this floorplanner to directly support physical-level thermal optimization and indirectly support architectural-level thermal optimization.

3.7.1. Floorplanner representation and cost function

The floorplanner within TAPHS is based on the Adjacent Constraint Graph (ACG) representation [**57**]. An ACG is a constraint graph with exactly one geometric relationship between every pair of modules. ACGs have invariant structural properties that allow the number of edges in the graph to be bounded. Operations on ACGs have straightforward meanings in physical space and change graph topology locally; they require few, if any, global changes. The operations of removing and splitting modules are designed to reflect high-level operation to functional unit binding decisions. To obtain the physical position of each module, packing based on longest path computation is employed. Simulated annealing is used to obtain an initial floorplan. A weighted sum of the area and the interconnect power consumption is calculated for use as the

floorplanner cost function, i.e.,

$$A + w \sum_{e \in E} C_e D_e \tag{3.28}$$

where A is the area, w is the power consumption weight, E is the set of all wires, e is an interconnect wire, C_e is the unit-length switched capacitance for the data transfer along e, and D_e is the length of e, which is calculated as Manhattan distance between the two modules connected by the wire. Using this cost function, we optimize the interconnect power consumption, interconnect delay, and area of the floorplan. The resulting floorplan will be improved during the subsequent incremental floorplanning high-level synthesis moves. Therefore, the number of simulated annealing iterations is bounded to reduce synthesis time.

After each high-level synthesis move, the previous floorplan is modified by removing or splitting a module. The modules and switched capacitances are updated based upon the impact of these merges and splits. The floorplan is then re-optimized with a greedy iterative improvement algorithm using the same cost function as the simulated annealing algorithm. There are two reasons to use a greedy algorithm during this stage of synthesis: (1) re-optimization requires fewer global changes and less hill climbing and (2) perturbations resulting from high temperatures may disrupt high-quality floorplan structures.

After determining the best binding across all the possible values of *csteps*, another simulated annealing floorplanning run is used for that binding. This final floorplanning stage occurs only once for every synthesis run. Therefore, it is acceptable to use a slower, but higher-quality, annealing schedule than those in the inner loop of high-level synthesis, thereby improving integrated circuit area and interconnect power consumption.

During the annealing schedule, we use a constant multiplicative cooling factor, r, i.e.,

$$T^{+} = r \times T \tag{3.29}$$

where T is the current temperature and $T+$ is the temperature during the next iteration. The number of the perturbations for the initial floorplanning run, the floorplanning for each value of *csteps* run, and the final floorplanning are related as follows: $1/2/20$. The number of perturbations per round for the greedy iterative improvement algorithm is the same as that for final floorplanning run.

3.7.2. Voltage island implementation in floorplanning

As described in previous section, voltage island generation was introduced into the high-level synthesis system in order to improve thermal profiles and reduce energy consumption. Therefore, the floorplanner must attempt to keep functional units assigned to the same voltage level contiguous in order to minimize the need for level converters and simplify power distribution. The floorplanner must still honor the elements in the original cost function shown in Equation 3.28. Pair-wise weighted edges were added between all pairs of functional units operating at the same voltage, yielding the following updated cost function:

$$n\sqrt{A} + 2n \sum_{v \in V} L_v + \sum_{e \in E} C_e D_e \tag{3.30}$$

where A is the area, n is the number of functional units, V is the set of all functional unit pairs at the same voltage, v is a pair of functional units the same voltage, L_v is the separation between a pair of functional units sharing the same voltage, E is the set of all interconnects, e, is an

interconnect line, C_e is the unit-length switched capacitance for the data transfer along e (zero in the case of no communication), and D_e is the length of e. This approach generates contiguous voltage islands, as well as optimizing the interconnect power consumption and area.

Figure 3.2, described in Section 3.3, shows an example of the results produced by this floorplanning algorithm. TAPHS rapidly generated this result using only pair-wise edges for functional unit clustering, i.e., hierarchical floorplanning was not required. Note that functional units operating at the same voltage are contiguous. In some cases, keeping voltage levels contiguous and minimizing wire length results in a slight area penalty. This is to be expected, regardless of the quality of a floorplanner, because it is rare for a minimal-area solution to have contiguous voltage levels and minimal interconnect power consumption. During incremental improvement, operation merging (functional unit resource sharing) combines functional units with other compatible functional units, always merging from the lower-voltage functional unit to the higher-voltage functional unit (please see Section 3.4).

3.8. Thermal modeling

As mentioned in Section 3.4, thermal modeling and analysis are used in the inner loop of the optimization flow to provide direct guidance for thermal optimization. Therefore, our previous work, a compact chip-package thermal model [**91**], has been integrated into TAPHS to determine the thermal profile of our system.

The thermal model has been validated against the COMSOL Multiphysics software package (formerly FEMLAB) [**92**], an accurate but slow commercial finite-element based solver. It exhibited less than 2.5% estimation error when measured on the Kelvin scale. In the following experiments, each chip design is attached to a copper heat sink using forced-air cooling. We

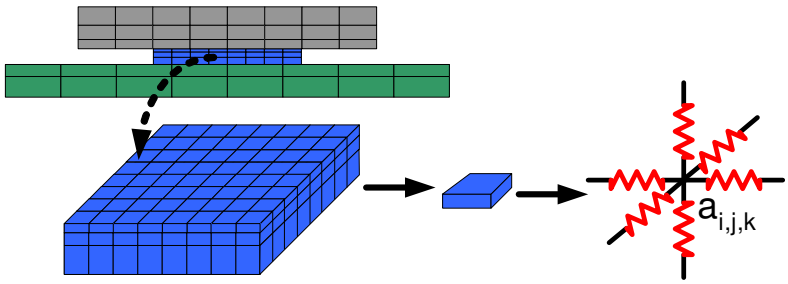


Figure 3.8. Full chip-packaging thermal model.

model two thermally conductive paths: heat dissipates from the silicon die through the cooling package to the ambient environment and through the package to the printed circuit board. We use an ambient temperature of 45 ℃ and a silicon thickness of 200 μm. In high-end microprocessor systems, due to the efficient cooling design, more than 80% of heat is dissipated through the first conductive path. In portable consumer electronics, due to the tight cooling budget and limited cooling space, the impact of the secondary conductive path may be significant.

Fig. 3.8 illustrates the compact thermal model. Each material layer (e.g., silicon die, cooling package, and substrate carrier) is modeled with multiple layers of thermal elements. Each layer is partitioned into homogeneous thermal tiles, and each thermal tile is then modeled with inter-layer and intra-layer thermal resistors. Thermal resistances are determined based on material properties and tile geometries. Different layers use different tile granularity to strike a good trade-off between estimation accuracy and efficiency. To estimate the power consumption of each tile, we currently ignore the self-heating effect of on-chip interconnect, and only consider the power consumption of active devices [**91**]. Note that this work considers the impact of wire capacitance on the power consumption of drivers and repeaters. Based on the floorplanning information, we compute the power consumptions of thermal tiles using the average power dissipated by the functional units within each thermal tile. The power consumption of

functional units with portions in multiple tiles is divided appropriately among the tiles. Instead of characterizing thermal profile from scratch after every incremental change to the power profile, the numerical thermal analysis method is initialized with the thermal profile associated with the previous power profile, thereby accelerating convergence after incremental changes to the power profile.

In general, chip thermal profile exhibits both temporal and spacial variations. Most of the benchmarks in this work have overall execution delays less than or comparable to the active layer element thermal time constant; therefore, temporal variation is negligible. Hence, we focus on characterizing spacial thermal variation using steady-state thermal analysis. Chip thermal characteristics are estimated based on the chip power distribution averaged over the period or the deadline for periodic benchmarks and aperiodic benchmarks respectively.

3.9. Experimental results

In this section, we present experimental results for the TAPHS temperature-aware high-level synthesis system, including the thermal optimization techniques described in Sections 3.5, 3.6, and 3.7. The circuits described in this section were synthesized using a register transfer level (RTL) design library based on the TSMC 0.18 µm process. The experiments were conducted on AMD Athlon-based Linux workstations with 512 MB–1 GB of random access memory. No IC synthesis runs required more than 1,195 s of CPU time.

3.9.1. Benchmarks

We used TAPHS to synthesize 13 synthesis benchmarks. *Chemical* and *IIR77* are infinite impulse response (IIR) filters used for signal processing. *DCT_IJPEG* is the Independent JPEG

Group's implementation of discrete cosine transform (DCT). *DCT_Wang* and *DCT_Lee* are DCT algorithms named after their inventors. All DCT algorithms work on 8×8 pixel of arrays. *Elliptic*, an elliptic wave filter, comes from the NCSU CBL (North Carolina State University Collaborative Benchmarking Laboratory) high-level synthesis benchmark suite [**64**]. *Jacobi* is the Jacobi iterative algorithm for solving a fourth-order linear system. *WDF* is a finite impulse response (FIR) wave digital filter. The largest benchmark, Jacobi, has 24 multiplications, 8 divisions, 8 additions, and 16 subtractions. In addition, we generated two CDFGs using a pseudo-random graph generator [**66**]. Random100 has 20 additions, 15 subtractions, and 19 multiplications. Random200 has 39 additions, 44 subtractions, and 36 multiplications. The same sample periods (deadlines) were used for the benchmarks when evaluating each synthesis technique.

3.9.2. Multiobjective results

Table 3.1 shows the results of doing full multiobjective optimization of peak temperature, area, and energy consumption with three voltage levels. In total, we compared 13 benchmarks. For each benchmark, the table shows non-dominated solutions produced by TAPHS. Due to space constraints, we sorted the solutions for each problem in order of increasing peak temperature and uniformly eliminated all but four solutions. For each solution, the left column indicates the name of the benchmark. The next three columns show the peak temperatures, areas, and power consumptions of solutions produced without using voltage islands. Area is reported as a percentage of the area of the an initial solution without resource sharing or voltage islands. The floorplanner typically has an area efficiency ranging from 75%–90% for these benchmarks. From these columns, it should be clear that it is possible to trade off peak temperature for area

Table 3.1. Comparison of non-dominated (multiobjective) results

Example	No voltage islands			Voltage islands		
	Peak T (°C)	Area (%)	Power (W)	Peak T (°C)	Area (%)	Power (W)
chemical	123.4	116.6	2.18	98.0	142.4	1.60
	123.6	112.0	2.18	100.4	121.7	1.62
	123.7	109.3	2.18	103.3	112.7	1.59
dct_dif	79.0	87.9	0.85	67.3	92.5	0.60
	79.7	78.6	0.83	67.6	81.5	0.58
	80.3	83.7	0.85	69.8	83.4	0.61
dct_ijpeg	126.0	118.2	2.44	113.6	117.6	1.99
	129.4	107.2	2.39	115.8	114.9	2.03
	129.5	104.5	2.41	118.6	99.9	2.00
dct_lee	71.5	98.9	0.79	63.7	106.4	0.59
	71.8	95.6	0.79	65.5	119.2	0.61
	71.9	99.9	0.79	64.6	106.3	0.59
dct_wang	70.7	101.3	0.70	59.8	109.8	0.42
	68.2	97.5	0.68	59.1	116.0	0.43
	68.5	108.1	0.68	60.1	108.0	0.42
elliptic	136.8	105.5	2.55	111.6	122.6	2.04
iir77	94.5	105.0	1.57	73.7	119.7	0.94
	97.7	93.1	1.56	74.6	115.7	0.94
	99.0	93.1	1.57	76.5	94.9	0.96
jacobi	54.2	64.4	0.25	51.8	81.5	0.20
	53.9	65.5	0.25	52.1	77.7	0.20
	53.8	63.2	0.24	52.9	69.2	0.21
pr1	98.0	104.0	1.49	82.5	106.1	1.10
	97.4	103.1	1.52	84.8	92.6	1.10
pr2	95.4	103.8	1.67	87.9	110.2	1.44
	97.3	89.2	1.68	87.5	100.6	1.45
	95.8	98.4	1.67	88.0	105.4	1.45
random100	71.6	100.0	0.85	66.0	98.8	0.63
	72.1	99.2	0.85	65.7	99.6	0.62
	72.7	99.7	0.86	67.6	85.1	0.67
random200	90.8	90.2	1.77	81.4	112.0	1.37
	91.1	93.0	1.77	83.2	90.2	1.37
wdf	75.6	108.0	0.75	68.0	104.5	0.59
	74.8	96.9	0.73	67.8	101.8	0.59

as long as a thermal model is available during multiobjective synthesis. However, improving both objectives requires architectural-level and physical-level thermal optimization techniques.

Table 3.2. Comparison of non-dominated (multiobjective) results with two voltage level

Example	No voltage islands			Voltage islands		
	Peak T (°C)	Area (%)	Power (W)	Peak T (°C)	Area (%)	Power (W)
chemical	123.4	116.6	2.18	104.1	113.3	1.64
	123.6	112.0	2.18	104.5	114.5	1.65
	123.7	109.3	2.18	106.0	117.9	1.70
dct_dif	79.0	87.9	0.85	71.5	81.2	0.62
	79.7	78.6	0.83	72.4	80.7	0.64
	80.3	83.7	0.85	71.9	77.7	0.60
dct_ijpeg	126.0	118.2	2.44	115.6	104.7	2.02
	129.4	107.2	2.39	118.2	107.9	2.08
	129.5	104.5	2.41	119.4	101.7	2.06
dct_lee	71.5	98.9	0.79	63.5	125.7	0.61
	71.8	95.6	0.79	65.1	97.6	0.61
	71.9	99.9	0.79	64.8	107.3	0.62
dct_wang	70.7	101.3	0.70	61.1	104.3	0.47
	68.2	97.5	0.68	60.2	96.2	0.46
	68.5	108.1	0.68	61.6	92.3	0.48
elliptic	136.8	105.5	2.55	116.6	112.3	2.09
	139.7	89.4	2.51	116.7	106.9	2.09
	141.0	105.5	2.48	118.6	106.9	2.09
iir77	94.5	105.0	1.57	77.5	102.4	1.01
jacobi	54.2	64.4	0.25	52.1	70.4	0.20
	53.9	65.5	0.25	52.9	68.2	0.20
	53.8	63.2	0.24	52.8	65.8	0.21
pr1	98.0	104.0	1.49	86.3	106.0	1.25
	97.4	103.1	1.52	88.0	102.6	1.28
	97.9	98.3	1.50	88.9	98.5	1.27
pr2	95.4	103.8	1.67	89.0	102.0	1.47
	97.3	89.2	1.68	89.7	95.7	1.47
	95.8	98.4	1.67	90.9	97.2	1.48
random100	71.6	100.0	0.85	64.8	108.2	0.64
	72.1	99.2	0.85	66.0	96.3	0.64
	72.7	99.7	0.86	65.9	100.6	0.64
random200	90.8	90.2	1.77	86.3	84.1	1.47
	91.1	93.0	1.77	85.6	82.8	1.48
	91.2	94.7	1.76	86.1	79.5	1.46
wdf	75.6	108.0	0.75	71.1	95.1	0.63
	74.8	96.9	0.73	73.0	84.0	0.64
	76.0	93.2	0.72	73.2	86.5	0.64

Table 3.3. Comparison of non-dominated (multiobjective) results with four voltage level

Example	No voltage islands			Voltage islands		
	Peak T (°C)	Area (%)	Power (W)	Peak T (°C)	Area (%)	Power (W)
chemical	123.4	116.6	2.18	96.9	131.5	1.55
	123.6	112.0	2.18	99.9	129.4	1.61
	123.7	109.3	2.18	101.8	116.1	1.56
dct_dif	79.0	87.9	0.85	66.8	105.7	0.59
	79.7	78.6	0.83	67.5	91.0	0.56
	80.3	83.7	0.85	66.8	94.8	0.57
dct_ijpeg	126.0	118.2	2.44	111.7	113.7	1.92
	129.4	107.2	2.39	115.8	116.4	2.05
	129.5	104.5	2.41	115.0	108.1	2.05
dct_lee	71.5	98.9	0.79	62.9	109.8	0.56
	71.8	95.6	0.79	63.9	109.6	0.59
	71.9	99.9	0.79	63.9	111.6	0.58
dct_wang	70.7	101.3	0.70	57.5	111.9	0.39
	68.2	97.5	0.68	57.6	112.3	0.40
elliptic	136.8	105.5	2.55	116.4	109.6	1.98
	139.7	89.4	2.51	115.5	104.8	1.98
	141.0	105.5	2.48	116.8	104.8	1.98
iir77	94.5	105.0	1.57	74.4	112.6	0.91
jacobi	54.2	64.4	0.25	52.3	67.3	0.20
	53.9	65.5	0.25	52.1	69.6	0.20
	53.8	63.2	0.24	53.2	66.2	0.21
pr1	98.0	104.0	1.49	80.3	107.6	1.07
	97.4	103.1	1.52	83.4	111.2	1.10
	97.9	98.3	1.50	84.1	101.4	1.08
pr2	95.4	103.8	1.67	86.7	106.9	1.44
	97.3	89.2	1.68	89.2	110.2	1.47
	95.8	98.4	1.67	88.4	99.7	1.44
random100	71.6	100.0	0.85	63.3	104.5	0.60
	72.1	99.2	0.85	64.3	98.4	0.60
	72.7	99.7	0.86	64.8	106.2	0.60
random200	90.8	90.2	1.77	80.5	94.4	1.35
	91.1	93.0	1.77	79.9	98.7	1.37
	91.2	94.7	1.76	80.7	94.9	1.38
wdf	75.6	108.0	0.75	71.1	95.1	0.63
	74.8	96.9	0.73	73.0	84.0	0.64
	76.0	93.2	0.72	73.2	86.5	0.64

The next three columns show the results produced using voltage islands. From these columns, it is clear that voltage islands yield significant improvements in peak temperature, area, and

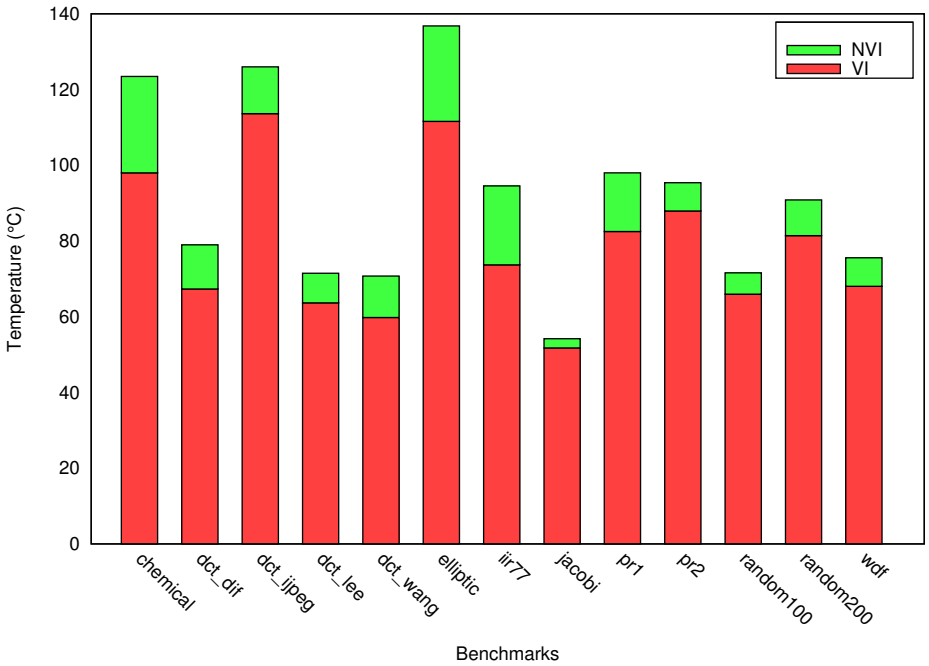


Figure 3.9. Peak temperature comparison for three voltage levels case.

power consumption. For example, the peak temperatures of the lowest peak temperature solutions to each problem were reduced by an average of 12.5 °C.

Figure 3.9 shows only the lowest peak temperature for each benchmark after synthesis with voltage islands, and without voltage islands. As this figure indicates that voltage islands can substantially reduce IC peak temperature, and that the relative contribution depends on the benchmark.

In addition, given the same area, TAPHS achieves lower peak temperatures for most benchmarks. For example, the peak temperature of *pr2* was reduced from 95.8 °C to 88.4 °C with the same area. Similar reductions were possible for *dct_dif*, *dct_ijpeg*, and *dct_lee*. In addition to reducing peak temperature, the proposed techniques can also be used to reduce area given a fixed peak temperature. When constraining temperature to the lowest temperature solution found

without thermal optimization techniques, using voltage islands reduced area by, on average, 9.9%.

As shown in Table 3.2 and Table 3.3, all the benchmarks were also run using two voltage levels and four voltage levels. The lowest peak temperatures were reduced by an average of 9.85 °C for two voltage levels, and 12.83 °C for four voltage levels, compared to the result without voltage islands. These results indicate that temperature reduction is significant when moving from two voltage levels to three voltage levels and minor from three voltage levels to four voltage levels. As shown in Figure 3.10, we also found that due to the differing properties (such as benchmarks size, CDFG graph structure, etc.) of each benchmark, the number of voltage levels permitting maximum temperature reduction differs. In general, temperature reduces with increasing voltage levels. However, changes in floorplanning prevent this trend from being entirely consistent.

3.10. Conclusions

In this chapter, we have described TAPHS, a thermal-aware high-level synthesis system that uses a tightly integrated thermal model and incremental floorplanner to optimize IC peak temperatures, areas, and power consumptions, while meeting performance constraints. In order to optimize peak temperature, it was necessary to tightly integrate floorplanning, wire modeling, power profile generation, and chip-package thermal analysis with high-level synthesis. Experimental results indicate that TAPHS is able to trade off peak temperature, IC area, and power consumption. The proposed techniques allowed a reduction in peak temperature of 12.5 ℃, on average. We have also found that thermal optimization can allow significant improvements in

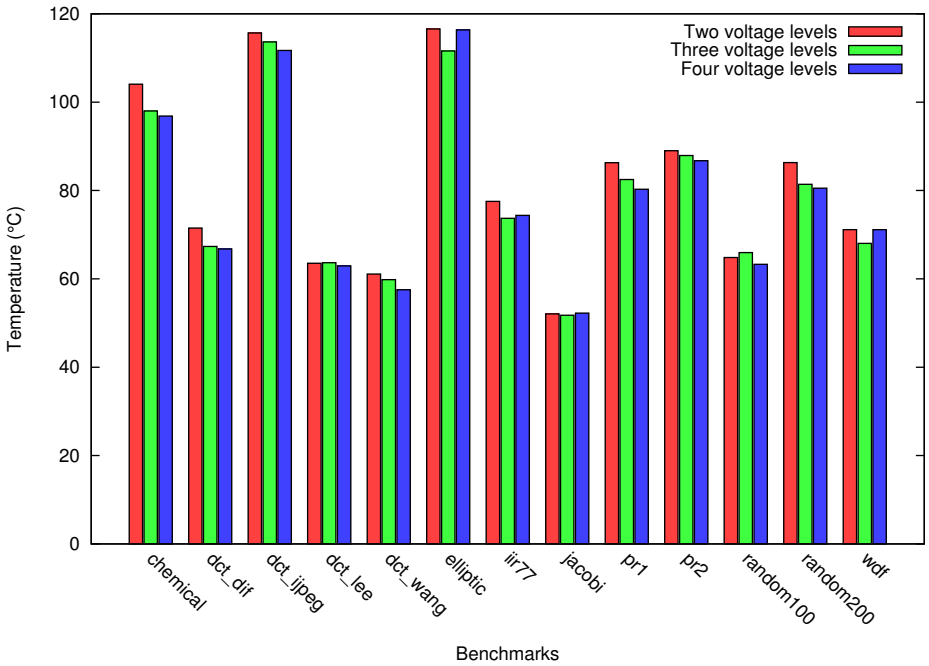


Figure 3.10. Peak temperature reduction with different number of voltage levels.

IC area under temperature constraints. We conclude that it is important to incorporate thermal optimization in high-level synthesis to support continued increases in device and power density.

CHAPTER 4

Reliable Application-Specific Multiprocessor System-On-Chip Synthesis

This chapter presents a comprehensive solution to the reliable multiprocessor system-on-chip (MPSoC) synthesis problem. Technology scaling and increasing power densities are increasing the severity of MPSoC lifetime reliability problems. MPSoC reliability strongly depends on system-level MPSoC architecture, the design style of each processing element (i.e., the presence or absence of redundant functional units), and the thermal profile during operation. We propose efficient thermal-aware reliability analysis and optimization algorithms for use during MPSoC synthesis. In addition, we have developed a comprehensive synthesis system, called RAMS, that conducts architectural synthesis, floorplanning, on-chip network synthesis, and chip-package thermal analysis. RAMS exploits redundancy and thermal-aware design planning to produce reliable, compact MPSoC designs that meet performance and functionality requirements. It is capable of substantially increasing MPSoC system mean time to failure with small area overhead. For example, it increases MPSoC system mean time to failure by an average of 85% with less than 5% area cost and by an average of 436% with less than 25% area cost, compared to area-optimized solutions.

4.1. Introduction

A single integrated circuit (IC) can now contain well over 500 million transistors. The International Technology Roadmap for Semiconductors (ITRS) [**2**] projects chips with a billion

transistors by 2007. In order to manage design complexity and control power consumption, it has been necessary to move to multiprocessor system-on-chips (MPSoCs).

Aggressive scaling of CMOS process technology poses serious challenges to the lifetime reliability of ICs. Reduction of feature sizes and increases in power density have resulted in increasing chip temperatures and failure rates. Increasing system integration using these vulnerable devices and interconnects results in reduced system reliability. The severities of many reliability problems, such as time-dependent dielectric breakdown in MOS transistors and electromigration in interconnects, increase exponentially with temperature. Lifetime reliability is becoming an important quality metric in high-performance ICs [**2**]. Optimizing lifetime reliability requires careful planning during IC design and synthesis.

The MPSoC lifetime reliability problem cannot be well solved at any single level of the design process. Reliability characterization requires detailed chip thermal information because the severities of many lifetime reliability problems strongly depend on temperature. Thermal characterization requires detailed physical information, including an IC floorplan, power profile, and chip-package thermal model. Reliability-aware MPSoC design requires a unified architectural-level and physical-level design flow. *Processing elements (PEs)* are general-purpose or special-purpose processor cores, many of which may be used in an MPSoC. At the architectural-level, careful assignment of tasks to PEs can balance the thermal profile of the chip, thereby improving system reliability. Synthesis-time architectural planning and careful use of PE-level and component-level (e.g., functional unit) redundancy will permit continued MPSoC operation after the failure of some processors or components, while limiting area overhead. At the physical-level, a fast floorplanner is needed to provide physical information for generating the

power profile which, in turn, is used to determine the thermal profile. The evaluation and optimization of system reliability and other design metrics, such as area and performance, require a comprehensive and efficient architectural-level and physical-level synthesis infrastructure.

4.1.1. Past work and Contributions

Our work draws from research in the areas of IC reliability modeling, reliable system synthesis, on-chip network synthesis, physical design, and thermal analysis. Srinivasan, et al., provide architectural reliability models and propose run-time techniques to improve the lifetime reliability of microprocessors [**93**]. The COFTA hardware-software co-synthesis algorithm produces architectures that achieve the reliability of triple-modular systems at lower cost [**94**]. Xie, et al., propose duplicating tasks on idle processors during embedded system synthesis in order to recover from transient faults [**95**]. Srinivasan et al. [**96**] propose a linear programming based NoC synthesis technique that maps a communication trace to an on-chip network topology to satisfy performance constraints and optimize power consumption. Ogras et al. [**97**] propose a depth-first search branch-and-bound algorithm for NoC synthesis.

IC temperature profile has a huge impact on reliability. In order to determine temperature profile, and conduct on-chip network synthesis, an MPSoC floorplan must be known during the system synthesis process. Research has shown that it is possible to incorporate high-performance floorplanning block placement algorithms within MPSoC synthesis [**98**, **8**]. We build upon IC thermal modeling research work in order to determine MPSoC thermal profiles to enable accurate estimation of system mean-time-to-failure (MTTF) [**10**, **99**].

The proposed algorithm generate a series of chip-level multiprocessor designs that satisfy the functionality and performance constraints of a specification while simultaneously trading

off and minimizing IC area and MTTF. The specification consists of graphs composed of data-dependent, multirate, periodic tasks and processor cores, each of which executes different tasks with potentially different execution times and power consumption values. It differs from previous work in the following ways:

(1) We provide a new, unified reliability model that supports estimation of the mean time an MPSoC will continue to meet its functional and performance requirements. This model takes various design-time and run-time factors into consideration, including numerous failure mechanisms, resource redundancy, and thermal profile and is efficient enough to permit repeated use during synthesis.

(2) We designed domain-specific optimization algorithms that significantly improve MPSoC reliability with small area overhead.

(3) We have developed a comprehensive MPSoC synthesis system that conducts architectural synthesis, floorplanning, on-chip network synthesis, and chip-package thermal analysis. Optimization algorithms within this flow exploit redundancy and thermal-aware design planning to produce reliable, compact MPSoC designs.

These ideas allow the proposed reliable MPSoC synthesis system to improve MPSoC system MTTF by an average of 85% with less than 5% area cost and by an average of 436% with less than 25% area cost, compared to area-optimized solutions. To the best of our knowledge, this is the first work that proposes a comprehensive temperature and lifetime reliability aware MPSoC synthesis system.

The rest of this chapter is organized as follows. Section 4.2 gives a motivating example that illustrates the need for reliable MPSoC synthesis and suggests techniques to optimize system MTTF. Section 4.3 describes the infrastructure, models, and optimization algorithms of our

reliable

application-specific multiprocessor synthesis system. Section 4.4 presents experimental results. We draw conclusions in Section 4.5.

4.2. MPSoC Reliability Estimation and Optimization Challenges

In this section, we illustrate the key challenges to estimating and optimizing MPSoC reliability. Section 4.2.1 defines system MTTF and uses an example to highlight the challenges of reliability optimization in MPSoC system design. Section 4.2.2 describes the challenges of calculating system MTTF. Finally, Section 4.2.3 describes the challenges of optimizing MPSoC system MTTF.

4.2.1. System MTTF Definition and Illustrative Example

System MTTF: *We define system MTTF as the average amount of time an MPSoC will operate, possibly in the presence of component faults, before its performance drops below some designer-specified constraint or it is no longer able to execute the specified workload.* Using system MTTF to characterize reliability has the advantage of taking into account performance; this is important for consumer electronics and most other MPSoC applications. It is often possible to increase reliability at the cost of reduced performance. This definition makes it clear that it is still sufficient to permit acceptable performance throughout the stated lifetime of the MPSoC.

In order to optimize the system MTTF of an MPSoC, while constraining IC area, it is necessary to exploit and optimize both hardware redundancy and temperature profile. *PE-level redundancy* is achieved by adding PEs to the MPSoC architecture. *Component-level redundancy*

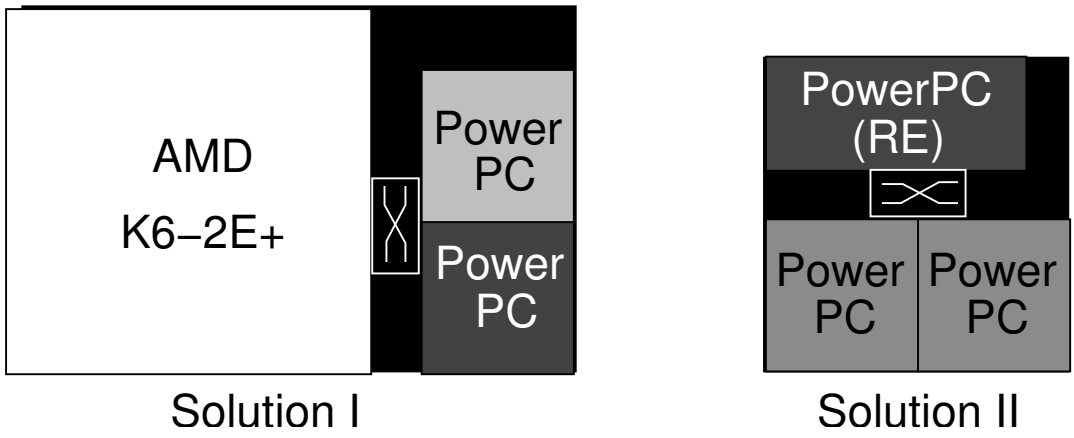


Figure 4.1. Reliable MPSoC synthesis example.

is achieved by adding appropriate control mechanisms and redundant hardware such as additional arithmetic logic units (ALUs) or cache banks to individual PEs [**93**]. We will illustrate each method of improving system MTTF using an example. Figure 4.1 shows two synthesized solutions of an MPSoC benchmark based on the E3S benchmarks suite [**100**]. Each solution contains three embedded processors connected by an on-chip router. The temperature of each on-chip component is indicated by its brightness: brighter components are hotter. The system MTTF of Solution I is 0.7 year. The system MTTF of Solution II in Figure 4.1 is 1.5 years, i.e., 2.1$\times$ the duration of Solution I. These two solutions differ as follows. The embedded processor, AMD K6-2E+, used in Solution I, is replaced with IBM PowerPC 405GP-RE in Solution II. 405GP-RE is a low power, redundant version of IBM PowerPC 405GP. In the 405GP-RE, the floating/fixed point units and floating/integer register files are duplicated.

In this example, the improvement to system MTTF in Solution II is mainly due to temperature reduction and resource redundancy. In ICs, many reliability problems strongly depend on temperature. In Solution I, a peak temperature of 59.9 °C is observed inside the K5-2E+. In Solution II, replacing the K5-2E+ with the 405GP-RE reduces the peak temperature by 5.1 °C,

thereby decreasing the run-time fault rate. Second, increasing system redundancy improves the system fault-tolerance. Compared to the K5-2E+, the 405GP-RE can tolerate more run-time faults. This results in an improvement to system MTTF. This example highlights various opportunities to optimize MPSoC reliability. However, synthesizing reliable MPSoCs is a challenging problem requiring efficient system MTTF calculation and a reliability-aware design optimization flow.

4.2.2. System MTTF Calculation Challenges

Computing system MTTF during MPSoC synthesis is a challenging task. Its complexity has two main sources: the difficulty of (1) determining the system-level impact of component failure and the necessity to synthesize an MPSoC architecture, from system level to physical level, and (2) computing its thermal profile in order to enable component failure probability estimation.

MPSoCs contain multiple heterogeneous on-chip components, or PEs. System MTTF is a complex function of component MTTFs. In a fault-tolerant MPSoC, a PE may have a partial failure or a complete failure that reduces MPSoC performance without causing the performance to drop below a specified value, i.e., MPSoC may still work even if some PEs suffer partial or complete failures. Assume each PE is in one of three status: fully-functional, partially operational [1], or inoperational. For an MPSoC system with N PEs there are 3^N possible combinations of processor states. Clearly, exact computation of system MTTF requires the traversal of a lattice with a number of points that is exponential in the number of processors. Although the

[1] In practice, a PE may experience various run-time failures, hence in different partially operational status.

traversal may terminate upon reaching a state in which the MPSoC does not meet its performance requirements, in the worst case a full traversal is necessary. Fast methods of computing system MTTF are essential during MPSoC synthesis because this computation will be used during the evaluation of numerous architectures.

Failure mechanisms in deep submicron application-specific multiprocessors are strongly dependent on MPSoC architecture, physical design, and thermal profile. Determining IC temperature profile requires knowledge of power profile which, in turn, requires knowledge of the power states and positions of PEs. Simply getting the necessary power information requires (extremely fast) generation of candidate architectural designs and their floorplanning block placements. Determining temperature profiles from the power profile requires (again, extremely fast) chip-package thermal modeling. Moreover, these models cannot be static: they must take into account conditions that may change dynamically during MPSoC operation and for each component, they must specify the type of failure (e.g., partial or complete) as well as the overall probability of failure.

4.2.3. System MTTF Optimization Challenges

Optimizing system MTTF requires that numerous hard, interdependent problems, many of which are NP-complete, be concurrently solved. These problems include PE allocation, the assignment of operations to PEs, floorplanning, on-chip network synthesis, operation and communication event scheduling, power consumption estimation, IC thermal analysis, component MTTF computation, and system MTTF computation. After each architectural change, the implications of the change on each level of the design must be quickly determined. In addition, it is necessary to compute system MTTF, which depends on pre-planning the appropriate run-time

task migration to compensate for a series of component failures while maintaining acceptable MPSoC performance. Using a traditional stochastic optimization algorithm by simply adding the system MTTF into the cost function as one of the optimization criteria will result in poor performance due to the high dimensionality of the multiobjective solution space. Therefore, it is necessary to develop domain-specific algorithms that can efficiently optimize MPSoC system MTTF and other criteria. As indicated by the experimental results in Section 4.4, a domain-specific algorithm that optimizes system MTTF by focusing on component redundancy and IC thermal profile is able to improve synthesis speed by an order of magnitude while increasing system MTTF by years without increasing area, in comparison with a stochastic algorithm that optimizes IC area and system MTTF.

4.3. Reliable Application-Specific MPSoC Synthesis

In this section, we present RAMS, the proposed reliable application-specific MPSoC synthesis infrastructure. Section 4.3.1 first gives an overview of RAMS. Section 4.3.4 then describes models for IC failure mechanisms. Section 4.3.5 proposes an MPSoC reliability model and corresponding MTTF-aware optimization algorithms used in RAMS. Finally, the thermal analysis (Section 4.3.6), floorplanning algorithms (Section 4.3.7), and on-chip network synthesis (Section 4.3.8) are presented.

4.3.1. RAMS Infrastructure

Determining and optimizing MPSoC system MTTF requires a substantial infrastructure, within which all the components indicated in Figure 4.2 are essential. As described in Section 4.2, computing system MTTF requires knowledge of component MTTFs and run-time performance constraints. Computing component MTTFs requires knowledge of MPSoC thermal

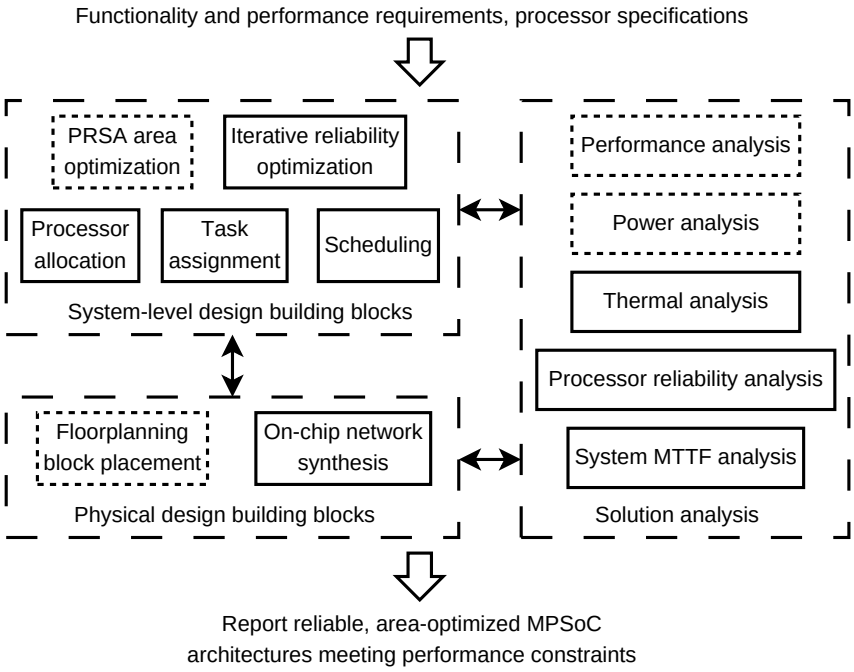


Figure 4.2. RAMS infrastructure for the synthesis of reliable MPSoCs taking into account architectural and physical effects on reliability.

profile and architecture. Computing MPSoC thermal profile requires fast and accurate thermal analysis algorithms. Finally, determining, and optimizing MPSoC architecture requires a system-level synthesis infrastructure that allocates PEs, assigns tasks to PEs, rapidly generates floorplans, synthesizes an on-chip network, assigns communication events to network links and routers, and schedules operations and communication events.

The RAMS synthesis system is composed of algorithms that may be placed in three categories: system-level design, physical-level design, and solution analysis. The system-level design contains a single-objective stochastic optimization algorithm that minimizes MPSoC area under performance constraints, and a heuristic optimization algorithm that uses knowledge of redundancy and thermal profile to improve system MTTF with little impact on MPSoC area.

The physical-level design consists of a slicing floorplanning algorithm and a novel MPSoC on-chip network synthesis algorithm. In addition, RAMS contains a novel statistical lifetime reliability model, and also performance, power, and thermal models to guide MPSoC reliability optimization.

4.3.2. System-Level Design

RAMS uses both stochastic and heuristic techniques to optimize MPSoC lifetime reliability at the system level.

The stochastic optimizer builds on our previous work, MOCSYN [**98**], a multi-objective SoC synthesis flow using a parallel recombinative simulated annealing (PRSA) algorithm [**101**].

Previous studies have demonstrates that the PRSA allocation, assignment, and scheduling algorithm used in MOCSYN produces optimal solutions to problems for which optimal solutions are known, and rapidly produces solutions of equal or better quality for problem instances to which results have been published [**102**, **103**]. The MPSoC lifetime reliability optimization issue can potentially be solved using this stochastic synthesis flow by combining lifetime reliability with the other optimization criteria supported by MOCSYN. However, compared to conventional area-performance optimization problems, reliability optimization introduces unique challenges.

In MPSoC design, the relationship between area and performance are complicated. Heuristic area and performance optimization techniques can easily be trapped in local minima because there is not a strong correlation between these objectives. Therefore, it is difficult to arrive at low-area solutions meeting hard real-time deadlines without a global optimization algorithm. Such area and reliability optimization heuristics stand in contrast to stochastic area and reliability optimization, which requires a global search of a multi-dimensional solution space. To guide

reliability optimization, reliability analysis needs to be invoked within the inner loop of the synthesis flow. MPSoC reliability analysis builds on detailed performance, power, and thermal characterization, which are computation intensive. Therefore, the use of reliability estimation within stochastic optimization dramatically decreases the area of the solution space that may be explored in a fixed amount of CPU time.

In comparison with the area-performance tradeoff, the area-reliability tradeoff is clearer. Lifetime reliability is inversely correlated with chip temperature and power density. By increasing chip area, power density and chip temperature decrease, thereby increasing chip reliability. Structural redundancy, which generally increases area, can also improve lifetime reliability. If it is possible to start from a low-area solution, the search for solutions that increase reliability at the cost of area can be focused on the most promising areas of the solution space via heuristics. The strong relationship between area and reliability permits the development and use of efficient heuristics for incremental MPSoC reliability and area optimization.

These observations motivate us to design a two-stage system-level optimization flow. This synthesis flow starts by searching the area-minimum solution using MOCSYN. Although area-efficient, these solutions tends to have high power density, low structural redundancy, and low lifetime reliability. These solutions are used as starting points for reliability optimization. The proposed heuristic reliability optimization algorithm uses knowledge of redundancy and thermal profile to improve system MTTF while minimizing impact on MPSoC area. This algorithm, one of the contributions of our work, is described in Section 4.3.5. As shown in the experimental results, the proposed two-stage optimization flow consistently outperforms one-stage stochastic area and reliability optimization.

4.3.3. Solution Analysis

MPSoC lifetime reliability is a function of various design-time and run-time factors, including failure mechanisms, resource redundancy, and chip thermal profile. In RAMS, MPSoC lifetime reliability optimization is guide by a novel MPSoC statistic lifetime reliability model that takes all these factors into consideration and provide efficient and accurate estimation of the system MTTF of the MPSoC.

The proposed reliability model builds on detailed performance, power, and thermal characterization. In RAMS, performance and power consumption are characterized for each individual on-chip PE based on measurement, datasheets, and discussions with microprocessor vendors (see Section 4.4 for details). As described in Sections 4.3.7 and 4.3.6, the impacts of physical characteristics, e.g., floorplans and thermal profiles, are considered and optimized by RAMS.

4.3.4. IC Failure Mechanisms

In this section, we characterize IC failure mechanisms. The lifetime reliability of ICs is primarily affected by the following failure mechanisms: electromigration, thermal cycling, time-dependent dielectric breakdown, and stress migration [**93**].

(1) **Electromigration** refers to the gradual displacement of the atoms in metal wires caused by electrical current. It leads to voids and hillocks within metal wires that result in open and short circuit failures. The MTTF due to electromigration is given by the following equation [**104**, **105**]:

$$MTTF_{EM} = \frac{A_{EM}}{J^n} e^{\frac{E_{a_{EM}}}{\kappa T}} \tag{4.1}$$

where A_{EM} is a constant determined by the physical characteristics of the metal interconnect, J is the current density, $E_{a_{EM}}$ is the activation energy of electromigration, n is an empirically-determined constant, κ is Boltzmann's constant, and T is the temperature.

(2) **Thermal cycling** refers to IC fatigue failures caused by thermal mismatch deformation. In IC chip and package, adjacent material layers such as copper/low-k dielectric have different coefficients of thermal expansion. As a result, run-time thermal variation causes fatigue deformation, leading to failures. The MTTF due to thermal cycling is given by the following equation [**104**, **106**]:

$$MTTF_{TC} = \frac{A_{TC}}{(T_{average} - T_{ambient})^q} \tag{4.2}$$

where A_{TC} is a constant coefficient, $T_{average}$ is the chip average run-time temperature, $T_{ambient}$ is the ambient temperature, and q is the Coffin-Manson exponent constant.

(3) **Time-dependent dielectric breakdown** refers to deterioration of the gate dielectric layer. This effect is a strong function of temperature, and is becoming increasingly prominent with the reduction of gate-oxide dielectric thickness and non-ideal supply voltage reduction. The MTTF due to time-dependent dielectric breakdown is given by the following equation [**104**, **93**]:

$$MTTF_{TDDB} = A_{TDDB} \left(\frac{1}{V}\right)^{(a-bT)} e^{\frac{A+B/T+CT}{\kappa T}} \tag{4.3}$$

where A_{TDDB} is a constant, V is the supply voltage, and a, b, A, B, and C are fitting parameters.

(4) **Stress migration** refers to the mass transportation of metal atoms in metal wires due to mechanical stress caused by thermal mismatch among metal and dielectric materials. The MTTF resulting from stress migration is given by the following equation [**104**]:

$$MTTF_{SM} = A_{SM}|T_0 - T|^{-n} e^{\frac{E_{a_{SM}}}{\kappa T}} \tag{4.4}$$

where A_{SM} is a constant, T_0 is the metal deposition temperature during fabrication, T is the run-time temperature of the metal layer, n is an empirically-determined constant, and $E_{a_{SM}}$ is the activation energy for stress migration.

Equations 4.1–4.4 indicate that the lifetime reliability of ICs is strongly influenced by temperature. Therefore, thermal analysis and optimization techniques play important roles in reliability optimization.

4.3.5. MPSoC Reliability Modeling and Optimization

The system MTTF of an MPSoC is a function of the lifetime reliability of all the on-chip PEs. In this work, we propose a unified lifetime reliability model for MPSoCs. Our first step is to derive an efficient modeling method that can accurately predict the lifetime reliability of each individual on-chip PE.

4.3.5.1. Reliability Modeling of On-Chip PEs. The lifetime reliability of an on-chip component is influenced by various design-time and run-time factors, such as architecture-level and circuit-level redundancy, run-time temperature, and fabrication process variation. In addition, different failure mechanisms have distinct run-time impacts. Accurate lifetime characterization of each component is challenging.

The PE reliability model used in RAMS considers fault mechanisms, component-level resource redundancy, and temperature. For the sake of explanation, in this section, the description of PE reliability modeling starts from the simplest case, i.e., a single failure mechanism, single point of failure (no resource redundancy), and constant temperature. These assumptions will later be relaxed, and the reliability model further generalized. Multiple failure mechanisms will first be considered, followed by component-level resource redundancy and thermal variation.

Statistical modeling is commonly used in IC reliability characterization. Researchers have proposed using various statistical models, e.g., exponential, Weibull, and lognormal, to characterize IC lifetime failures. Compared to other commonly-considered statistical models, the lognormal distribution more accurately models the time-dependent degradation processes of ICs, e.g., diffusion, corrosion, migration, and crack propagation [**93**] caused by the failure mechanisms described in Section 4.3.4. However, using the lognormal distribution complicates the derivation of analytical solutions [**107**]. Numerical methods, such as Monte-Carlo simulation or statistical fitting techniques, are required. These methods are computational intensive.

Starting from the simplest assumption, for a failure mechanism i, the run-time fault probability density function (PDF), $f_i(t)$, and the corresponding cumulative fault distribution function (CDF), $F_i(t)$, can be described with two parameters: σ^i_{PE} (a shape parameter) and μ^i_{PE} (a scale parameter). The MTTF of an on-chip PE due to a particular failure mechanism i, $MTTF^i_{PE}$, is then estimated as follows:

$$MTTF^i_{PE} = \int_0^{\infty} t\ f_i(t)dt = \int_0^1 t\ dF_i(t) = e^{\mu^i_{PE} + {\sigma^i_{PE}}^2/2} \tag{4.5}$$

The overall lifetime reliability of each on-chip PE, $MTTF_{PE}$, is modeled by a joint lognormal distribution that depends on the major failure mechanisms described in Section 4.3.4. We assume that the relationships among different failure mechanisms are serial, i.e., each individual failure mechanism can result in complete PE failure independently. Therefore, for each non-redundant PE, the CDF of its overall lifetime failure probability follows:

$$F_{PE}(t) = 1 - \prod_i (1 - F_i(t)) \tag{4.6}$$

where i is the index of different failure mechanisms.

Researchers have often used exponential distributions for statistical modeling due to their convenience. Given $F_i(t)$s with exponential distributions, Equation 4.6 would yield an easily-computed analytical solution. However, as a consequence of using the more accurate lognormal distribution for each $F_i(t)$, Equation 4.6 does not allow straight-forward estimation of PE MTTF, $MTTF_{PE}$. In this work, we use statistical fitting to approximate $MTTF_{PE}$ using a single lognormal distribution, governed by μ_{PE} and σ_{PE}. The parameters for this approximation follow:

$$\mu_{PE} = \frac{1}{2} \log \left(\frac{\left(\int_0^\infty t \; dF_{PE}(t)\right)^4}{\int_0^\infty t^2 \; dF_{PE}(t)} \right) \tag{4.7}$$

$$\sigma_{PE} = \sqrt{\log \left(\frac{\int_0^\infty t^2 \; dF_{PE}(t)}{\left(\int_0^\infty t \; dF_{PE}(t)\right)^2} \right)} \tag{4.8}$$

As described in Section 4.2.1, PEs may have component redundancy to improve performance or reliability. Such PEs can be designed to continue functioning even if some of their components, e.g., an ALU or a cache bank, fail. By taking resource redundancy into consideration, the lifetime reliabilities of PEs can be characterized as follows.

Assume a PE contains M types of resources. Each type of resource $S_i, i \in \{1, \cdots, M\}$, is comprised of N_i identical elements. Assume the cumulative failure probability of resource element $E_{i,j}, i \in \{1, \cdots, M\}, j \in \{1, \cdots, N_i\}$ is $F_{i,j}(t)$. Then, the cumulative failure probability of resource S_i, $F_{S_i}(t) = \prod_j F_{i,j}(t)$. The MIN–MAX model may be used to bound the MTTF of a PE with M types of resources as follows:

$$MTTF_{PE} = \min_{i=1}^{M} \left(\int_0^1 t \; dF_{S_i}(t) \right) \tag{4.9}$$

Note that, run-time faults will generally result in performance degradation, even if the affected PE continues to function. Many applications have performance requirements. If, as a result of PE performance degradation, the MPSoC performance drops below some bound, the MPSoC should be deemed to have failed. Detailed MPSoC system reliability analysis under MPSoC performance requirements is described in Section 4.3.5.2.

The lifetime reliability of a PE strongly depends on its temperature. In RAMS, after each MPSoC solution is derived, performance and power analysis are conducted. The estimated power profile and MPSoC floorplan, as well as the chip, package, and cooling configuration, are provided to a thermal analysis algorithm to determine the chip thermal profile. Note that Equation 4.9 is derived under an assumption of constant PE temperature. Next, we discuss how to estimate temperature-dependent PE MTTF.

The temperature profile of an MPSoC varies with run-time workload. The impact of temperature variation on MTTF calculation is illustrated in Figure 4.3. In this example, T_1 and T_2 are temperatures. T_1 is high and T_2 is low. The PE is initially hot (T_1) and, at time t_1, becomes cooler (T_2). Functions $f_1(t)$ and $f_2(t)$ are the fault PDFs given temperatures T_1 and T_2, respectively. The overall fault distribution of the PE should satisfy the following equation, i.e., the

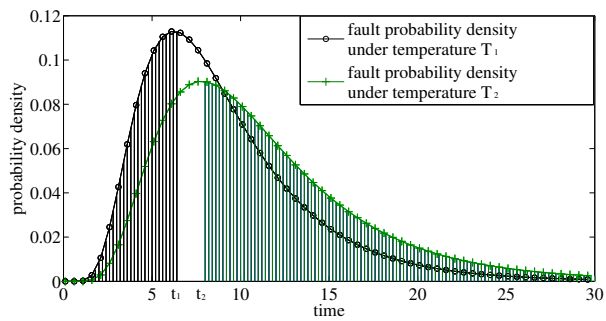


Figure 4.3. Temperature impact on MTTF.

overall cumulative fault distribution equals one.

$$\int_0^{t_1} f_1(t)dt + \int_{t_2}^{\infty} f_2(t)dt = 1 \tag{4.10}$$

When we switch from the fault PDF associated with one temperature, e.g., T_1, to that associated with another temperature, e.g., T_2, it is necessary to adjust our start time to the value, in the new time scale, associated with the appropriate amount of wear that had been experienced in the previous time scale, i.e., we must start integrating from the effective age of the PE. For this example the concept can be summarized as follows: $F_1(t_1) = F_2(t_2)$.

Given that $\{T_0, T_1, \cdots, T_{N-1}\}$ denote the run-time PE thermal profile, the overall fault distribution should satisfy the following equation:

$$\int_{t_{s0}=0}^{t_{e0}} f_0(t)dt + \int_{t_{s1}}^{t_{e1}} f_1(t)dt + \cdots + \int_{t_{sN-1}}^{\infty} f_{N-1}(t)dt = 1 \tag{4.11}$$

where $f_i(t)$ denotes the fault PDF of the PE at temperature T_i, $t_{ei}(t)$ denotes the transition time at which the temperature changes from T_{i-1} to T_i, and $t_{si}(t)$ denotes the equivalent age of the PE, starting from t_{ei-1}, when the temperature switches to T_i. The value of t_{si} can be determined

using Equation 4.11, allowing the MTTF of a PE to be determined by the following equation:

$$MTTF = \sum_{i=0}^{N-1} \int_{t_{si}}^{t_{ei}} t f_i(t) dt \tag{4.12}$$

In RAMS, reliability analysis resides in the inner loop of system synthesis. Therefore, modeling efficiency is critical. An MPSoC consists of numerous on-chip PEs. If the cumulative fault probability distributions, $F_i(t)$, are lognormal, then solving Equation 4.9 requires computationally-intensive numerical analysis. To address this issue, we produce a PE reliability library before synthesis by pre-characterizing the reliability distributions of PEs as functions of temperature and supply voltage. During MPSoC synthesis, when solving Equation 4.12, the value of $F_i(t)$ can be obtained by table look-up.

4.3.5.2. Reliability Modeling and Optimization of MPSoCs. In this section, we discuss MPSoC lifetime reliability estimation and optimization.

Many MPSoCs have built-in resource redundancy. In the recent past, techniques to provide both component-level (intra-PE) and PE-level redundancy have been proposed to improve system reliability and performance (see Section 4.2.1 for definitions). For MPSoCs with resource redundancy, run-time faults may or may not cause system failures. As highlighted in Section 4.2, in RAMS, system MTTF is defined as the mean operating time until the system fails to meet designer-specified functionality and performance requirements.

The reliability analysis and optimization flow is shown in Figure 4.2. In RAMS, reliability optimization starts by evaluating the system MTTF of area optimized solutions (using Algorithm 4), which tend to have high power density, high temperature, low resource redundancy and, therefore, low system MTTF. An iterative reliability optimization algorithm is invoked

if these solutions cannot meet targeted system MTTF. During each iteration, Algorithm 5 optimizes the system MTTF by improving system resource redundancy and/or optimizing chip thermal profile by introducing new PEs and/or replacing or reinforcing vulnerable PEs. System-level (task assignment and scheduling) and physical-level (floorplanning and network synthesis) algorithms are then invoked to produce valid MPSoC solutions. Through performance, power, thermal, and reliability analyses, the system MTTF of new solutions are estimated and evaluated. The iterative optimization flow continues until the targeted system MTTF is achieved.

Algorithm 4 estimates system MTTF based on statistical models of MPSoC run-time failure processes. Starting from time $t = 0$, it determines the minimal MTTF among all the on-chip PEs using Equation 4.12 (line 4). Each fault may result in partial or complete PE failure. In either case, task migration is used to balance system workload and optimize system performance. The task migration routine moves tasks from the faulty or partial faulty, and therefore lower performance, PE to other PEs (line 6). After task migration, if the MPSoC still meets its performance requirements, the algorithm moves on to the remaining fault implying the minimal MTTF. Task migration results in run-time changes in chip power consumption and temperature profiles, thereby changing the lifetime reliability of each on-chip PE. To accurately predict subsequent PE MTTFs, power and thermal analysis are conducted (lines 8 and 9). This process continues until the MPSoC fails to meet its performance or functionality requirements. The system MTTF of the MPSoC solution is then reported (line 11).

At run-time, on-line fault detection algorithms will be used to determine when an execution unit has failed. A proper treatment of on-line fault detection is beyond the scope of this work but can be found in the literature [**108**, **109**]. Upon fault detection, the pre-planned task assignment changes indicated by the migration associated with the particular fault are made. If

Algorithm 4 System MTTF analysis of an MPSoC solution

1: Given an MPSoC solution $MPSOC_{Sol}$, $MTTF_{MPSoC} = 0$, time $t = 0$
2: **while** system schedule is valid **do**
3: $MPSoC_{Func}$ are the functioning PEs in $MPSoC_{Sol}$
4: Fault interval $e_i = \min_{pe \in MPSoC_{Func}}(MTTF_{pe})$
5: $MTTF_{MPSoC} += e_i$
6: Task migration, scheduling
7: **if** system scheduling is valid **then**
8: Power analysis, thermal analysis
9: Determine the temperature of each PE
10: **else**
11: Return $MTTF_{MPSoC}$
12: **end if**
13: **end while**

it is acceptable to reboot the system in the presence of a fault, no further provisions need be made. Note that a system is likely to experience only a few faults over its MTTF lifespan, e.g., 10 years. If uninterrupted operation in the presence of a fault is necessary, distributed system checkpointing will suffice [**110**].

RAMS is equipped with an efficient workload migration algorithm to maintain system functionality and meet performance requirements in the presence of partial and complete PE failures. When an MPSoC fails to meet its performance requirements due to run-time faults, tasks originally assigned to the faulty PE migrate to other PEs using the following policy. Tasks on faulty PEs are first sorted in order of increasing time slack, the difference between the task's latest finish time and earliest finish time. They are then migrated from the PE, to other PEs, in this order until the system performance requirements are met and no tasks are assigned to a totally failed PE. When moving a task from one PE to another, the new PE is selected by Pareto-ranking [**111**] all PEs in order of increasing utilization ratio (the proportion of time during which the PE is actively executing tasks) and increasing execution time for the task and PE under consideration.

Note that, if the faulty PE is inoperational, then all the tasks assigned to it migrate. If the PE has only partially failed, only a subset of its tasks migrate to other PEs.

RAMS optimizes the lifetime reliability of MPSoCs by focusing on architectural changes that improve redundancy and thermal profile, while maintaining low area overhead. Algorithm 5 shows the actions taken by RAMS to improve an MPSoC architecture that does not have a sufficient system MTTF, i.e., $MTTF_{MPSoC} < MTTF_{target}$. First, the MTTF of each individual PE is estimated (line 2). The PE with the minimal MTTF is identified as the MPSoC's most vulnerable point, PE_{vul} (line 3). RAMS then attempts to improve system MTTF via one of our proposed reliability optimization moves: PE reinforcement, PE swapping, and PE addition (line 4). *PE reinforcement* introduces component redundancy (see Section 4.2.1) into the most vulnerable PE. *PE swapping* replaces the most vulnerable PE with a different, more reliable, PE. *PE addition* introduces a new PE into the MPSoC, enabling tasks to migrate from the vulnerable PE to other PEs. These moves consider multiple candidates PEs. RAMS uses the relative reliability gain, defined in Equation 4.13, to select the best candidate move. This equation takes both power density reduction, resource redundancy improvement, and area overhead associated with the move into consideration.

$$G_{RAMS} = \frac{e^{-P_d} \times MTTF_{ref}}{A} \tag{4.13}$$

Note that this value is used only to guide changes. The detailed effect of each tentative change is computed using thermal profile and reliability analysis. MPSoC power profile influences MPSoC temperature profile, which strongly influences reliability. The MTTFs associated with some major fault mechanisms are exponential functions of temperature. Therefore, in

Equation 4.13, RAMS uses an exponential term, e^{-P_d}, to characterize the impact of power density reduction on reliability improvement. P_d is the power density reduction resulting from applying a candidate move. In Equation 4.13, the impact of redundancy is characterized by the second term, $MTTF_{ref}$, the system MTTF improvement resulting from the candidate move. $MTTF_{ref}$ is calculated under the assumption that other design characteristics, e.g., temperature profile and supply voltage, remain the same. The relative reliability gain introduced by each candidate move is the product of these two terms divided by the area overhead. The move with the highest gain is applied (line 5). After each optimization move, system-level and physical-level synthesis algorithms are invoked to update the MPSoC solution. Cost analysis is then conducted to determine the improvement in system reliability, determine the impact on MPSoC area, and validate the system schedule. This optimization process continues until the target system MTTF is achieved.

For comparison purposes, we also implemented two other optimization moves. The first considers only power density, e^{-P_d}, and the second considers only resource redundancy, $MTTF_{ref}$. Performance comparisons among these three heuristics are provided in Section 4.4.

4.3.6. Thermal Analysis

Our interest in the synthesis of reliable ICs motivated us to develop a new thermal analysis algorithm with the accuracy and speed to support this application. Existing IC thermal analysis tools [**76**, **73**, **92**] are capable of providing either accuracy or speed, but not both. Accurate thermal analysis requires expensive computation for many elements in some regions, at some times. Academic IC thermal analysis techniques ensure accuracy by choosing uniformly fine levels of detail across time and space, i.e., they use equivalent physical sizes or time step durations for all

Algorithm 5 Reliability-aware optimization algorithm

```
1:  while MTTF_MPSoC < MTTF_target do
2:      ∀_{pe∈MPSoC} compute MTTF_pe
3:      Find vulnerable point: PE_vul is the PE with minimal MTTF
4:      Optimization moves (PE reinforcement, PE swapping, PE addition)
5:      Apply the best move based on Equation 4.13
6:      System-level synthesis
7:        Task assignment
8:        Scheduling
9:      Physical-level synthesis
10:       Floorplanning
11:       On-chip network synthesis
12:     Performance, power, thermal, reliability analysis
13:     if system MTTF improves and system schedule is valid then
14:         Continue
15:     else
16:         Revert this change
17:     end if
18: end while
```

thermal elements. The large number of elements and time steps resulting from such techniques makes them computationally intensive and, therefore, impractical for use within IC synthesis.

We have developed an accurate, spatially and temporally adaptive chip-package thermal analysis tool [**10**] for use within IC synthesis algorithms. When used for steady-state thermal analysis, it takes, as input, a three-dimensional chip and package thermal conductivity profile, as well as a power dissipation profile. A multi-grid incremental solver is used to rapidly produce an IC temperature profile. Heterogeneous, problem instance specific, adaptation of spatial resolution is used to dramatically reduce computational overhead without sacrificing accuracy. Recall that high-accuracy thermal profile estimation is required for reliability estimation because the severities of many reliability problems are exponentially dependent on temperature.

We validated our proposed thermal model using COMSOL Multiphysics [**92**], a reliable commercial finite element physical process modeling package. When used on academic and

commercial IC designs, the worst-case errors for the proposed thermal analysis algorithms are less then 1.7% on the Celsius scale and less than 1% on the Kelvin scale. In order to support IC synthesis, thermal analysis must be fast enough to allow numerous evaluations in the inner loop of a synthesis flow. The proposed technique is 22×–690× faster than a similar multi-grid technique that does not do heterogeneous element adaptation [**73**], and at least 15,000× faster than COMSOL Multiphysics run on the same problems.

4.3.7. Floorplanning

This section summarizes the area and communication time aware floorplanning algorithm designed to rapidly determine the impact of MPSoC architectural changes on design metrics such as area, performance, and thermal profile. Initially, the shape of each PE is determined by two datum: the raw intellectual property PE layout shapes that are specified in the resource database and the area of memory required by each PE, which must be determined during the synthesis process. The memory requirement of each PE depends on the tasks assigned to it. After PE areas have been calculated, an area-balanced binary tree of PEs is formed based on *tie priorities*, i.e., the priorities of communication between PEs pairs. Accounting for the priority of communication is an extension of an algorithm designed by Fiduccia and Mattheyses, which considered only the presence or absence of communication [**58**]. Considering priority increases the time complexity of the partitioning algorithm from $O\left(n^2\right)$ to $O\left(n^2 \cdot \log n\right)$ where n is the number of PEs. PEs that are adjacent in the binary tree are also adjacent in the final block placement. After forming the binary tree, RAMS optimally determines the orientations of all of the PEs, under the constraint that the aspect ratio, i.e., the ratio between width and height, does not exceed a value specified by the user. This algorithm is based on past work [**59**]. It takes

$O(n \cdot \log n)$ time where n is the number of PEs when PE orientations, but not PE rotations, are determined. The problem of optimally determining optimal PE rotations and orientations is NP-complete [**112**]. However, for the number of PEs typically encountered in SoC synthesis problems (fewer than 200), a natural extension of Stockmeyer's work [**59**] to optimal rotation and orientation determination has acceptable run-time in practice.

4.3.8. On-Chip Network Synthesis Algorithm

This section describes our heterogeneous on-chip network synthesis algorithm. This algorithm generates a heterogeneous on-chip network of processor cores with differing shapes by taking physical information into consideration. The input of the algorithm is a communication graph, $G = (V, E)$, in which V is the set of PEs allocated by RAMS and E is a set of weighted edges. Each edge, $e_i \in E$, has a weight equal to the *communication timing slack* between the connected pair of PEs, which is defined as the difference between latest start time and earliest start time.

As shown in Algorithm 6, the communication graph, G, is taken as input. The physical positions of the PEs are generated by our floorplanner, which is described in Section 4.3.7. Here, we define *communication density* as the communication quantity per unit area. This variable indicates the amount of data that passes through a region. The communication density profile of a chip is generated (Algorithm 6 line 1) in the following way: communication density of region CD_i is the sum of all the communication events of each pair of PEs that may transfer data through region i, i.e.,

$$CD_i = \sum_{v \in \text{region } i} v/a \tag{4.14}$$

All regions are sorted in order of decreasing communication density (Algorithm 6 line 2). A *router active region* is defined to be a square (due to orthogonal routing), centered on the router. Communication events passing through a router's region may be forwarded by the router. Routers are inserted starting from the most dense region, reducing communication density to zero within the router active region (Algorithm 6 line 3 to 7). Ideally, the physical position of router r is the center point of all the PEs associated with r. However, this is impractical in on-chip network synthesis for heterogeneous PEs since the insertion point may already be occupied by a PE. Hence, routers are inserted along the shared boundaries of their associated PEs. We exploit the slicing tree structure used for PEs floorplanning to determine the position of the router r. Note that in the slicing tree structure of the floorplanner, each leaf node represents a PE. Therefore, the position of the router may be determined by finding the lowest-level (possibly transitive) parent node of all the leaf nodes within the router's active region. All the PEs that have not yet been associated with other routers are now connected to the newly-inserted router, r, as long as they are within the active region of r. This procedure will be conducted repeatedly until no regions with communication density greater than 0 remain. At this point routers are connected to form a triangular mesh (Algorithm 6 line 8). The final steps (Algorithm 6 line 9 to 12) map the communication events into routing paths based on a weighted sum of communication quantity and slack. This causes communication events with less slack to be mapped to paths with fewer routers, thereby reducing latency. Here the slicing tree structure is reused to determine the routing path of each communication event between source and destination PEs.

4.4. Experimental Results

This section first describes the benchmarks used to evaluate RAMS. Section 4.4.2 shows the results produced by running RAMS, and an area-minimizing stochastic optimization algorithm,

Algorithm 6 Heterogeneous on-chip network synthesis algorithm (G)

```
Generate communication density profile
Sort in decreasing order of communication density
while there exists area with density greater than 0 do
    Insert a router r in the current highest density area
    Connect r to all the un-associated PEs within the active region of router r
    Set the density of the area covered by router r to be 0
end while
Connect routers to form a triangular mesh
Sort the communication events based on the weighted sum of the communication volume and slack
while there exists unmapped communication event do
    Map the communication event by using slicing tree structure to find the route path
end while
```

when run on these benchmarks. RAMS increases MPSoC MTTF by an average of 85% with less than 5% area cost or by an average of 436% with less than 25% area cost. Section 4.4.3 describes the impact of each of our proposed optimization moves on MPSoC system MTTF and area; we conclude that both redundancy and thermal profile should be optimized during reliable MPSoC synthesis. Note that, as a result of the definition of system MTTF in Section 4.2.1, we only admit solutions in which all tasks meet their deadlines, i.e., we do not consider solutions with acceptable performance to be valid. Section 4.4.4 compares the quality and performance of RAMS with a multiobjective stochastic optimization algorithm that maximizes system MTTF but does not use domain-specific optimization moves; we conclude that the proposed hybrid optimization flow improves the efficiency and quality of results for nearly all problem instances.

4.4.1. Benchmarks

The proposed reliable MPSoC synthesis algorithm was evaluated using a number of benchmarks based on the E3S benchmarks suite. E3S contains 17 PEs, e.g., the AMD ElanSC520, Analog Devices 21065L, the Motorola MPC555, and the Texas Instruments TMS320C6203.

These PEs are characterized based on the measured execution times of 47 tasks commonly encountered in embedded applications, power numbers derived from datasheets, and additional information, e.g., PE areas, some of which were necessarily estimated, and prices gathered by emailing and calling vendors. Any PE for which the datasheet reflected results in coarser technologies were linearly scaled to a 0.18 μm technology. The E3S task sets follow the organization of the EEMBC benchmarks [**113**]. There is one task set for each of the five application suites: Automotive/Industrial, Consumer, Networking, Office Automation, and Telecommunications. The Office Automation problem contains only five tasks. Our modified version of Office Automation contains four copies of the original Office Automation task set. In addition, TGFF [**66**] was used to generate five random benchmarks, each of which has 30–50 tasks. The graphs have different structures, ranging from random connectivity to a series-parallel structure commonly encountered in DSP applications. For each random benchmark, every task was randomly assigned a task type from E3S.

In E3S, PEs do not have component redundancy, i.e., each PE will fail if any of its functional units fails. We introduce a redundant version for each PE in E3S by duplicating floating/fixed point units and floating/integer register files. Following previous work [**93**], we assume that instruction scheduling units and instruction decode units do not have redundancy, i.e., a single run-time fault in these units will result in PE failure. On-chip caches have redundancy, i.e., a single fault will reduce performance but the PE will remain operational. This redundancy comes at a cost. We relied on previous work to estimate the side-effects of component redundancy [**93**]: PEs with component redundancy suffer a 24% area penalty and, as long as their additional functional units are still operational, have 25% higher performance and power consumption.

The PEs in E3S have fairly homogeneous performance. This is not surprising given their intended use: stand-alone embedded microprocessors or microcontrollers. It is our goal to develop a synthesis algorithm that is effective at improving the reliability of application-specific MPSoCs, which commonly contain heterogeneous PEs. The energy-delay products of PEs in E3S are fairly uniform. Therefore, for each PE in E3S, we introduced one corresponding PE operating at a higher voltage and another operating at a lower voltage. Note that a maximum of three voltages need be provided by off-chip regulators. The alpha power law was used to calculate the impact of voltage scaling on performance. Recall that we are considering a 0.18 μm process in these benchmarks. A nominal supply voltage of 1.8 V and alpha of 1.3 were used, based on recent short-channel MOSFET characteristics [**114**]. Therefore, to model high-performance PEs, the supply voltage was scaled to 2.5 V, the performance increased by 25%, and the power consumption increased to 2.4×. To model low-power PEs, the supply voltage was scaled to 1.28 V, performance was increased by 25%, and power consumption was decreased to 0.38×.

4.4.2. Comparison of RAMS and Stochastic Area Optimization

As described in Section 4.3.1, RAMS consists of a two-stage optimization flow. MOCSYN PRSA algorithm first optimizes MPSoC area under performance requirement. Although the MTTF of the solution has not been optimized, its area is optimized and real-time deadlines are honored. The produced area-optimized solution is then used as a starting point for the MTTF optimization heuristics.

Figure 4.4 illustrates the solutions produced by the RAMS MTTF optimization technique for all ten benchmarks. In this figure, for each benchmark, the initial solution produced by PRSA appears at the left-most point of the line associated with the benchmark. We continued to

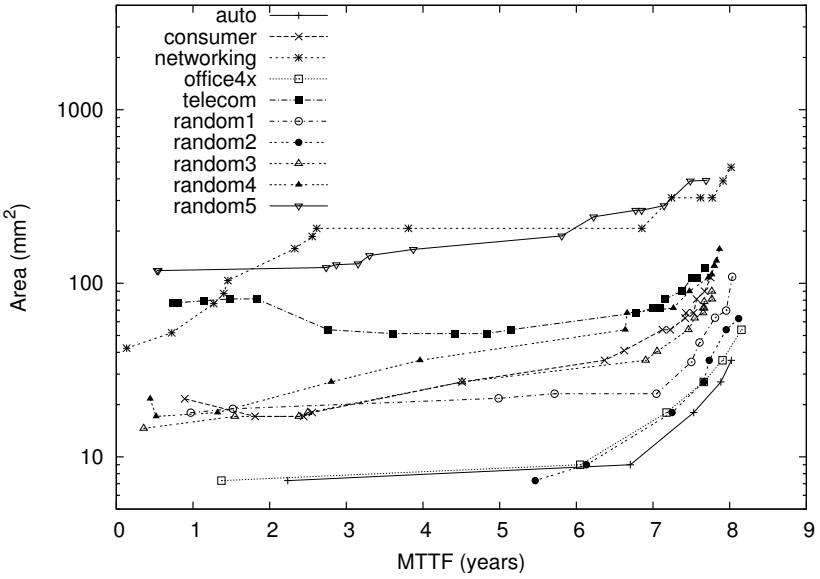


Figure 4.4. Solutions produced by RAMS.

Table 4.1. System MTTF improvement under area bound

Area bound (%)	MTTF improvement (%)	Area bound (%)	MTTF improvement (%)	Area bound (%)	MTTF improvement (%)
0.0	40.0	15.0	180.0	30.0	457.0
5.0	85.0	20.0	240.0	35.0	468.0
10.0	180.0	25.0	436.0	40.0	470.0

The MTTF improvement under each area bound is computed by selecting the highest-MTTF solution, for each benchmark, that honors the area bound and computing the average of their MTTF improvements.

apply the optimization moves described in Section 4.3.5.2 until seven subsequent moves did not significantly improve system MTTF. Table 4.1 shows the average system MTTF improvement over initial area-optimized solutions under different area overhead constraints for all ten benchmarks. These results illustrate three key points about the reliable application-specific MPSoC synthesis problem.

Table 4.2. MTTF improvement with 25% area overhead bound

Benchmarks	MTTF			Area		
	Init (years)	Final (years)	Improve (%)	Init (mm^2)	Final (mm^2)	Overhead (%)
auto	2.2	6.7	199.8	7.3	9.0	23.5
consumer	0.9	4.5	405.2	21.6	27.0	25.0
office4x	1.4	6.1	341.1	7.3	9.0	23.5
networking	0.1	0.7	435.9	42.3	51.8	22.7
telecom	0.7	7.4	895.8	77.4	90.0	16.4
random1	1.0	5.0	413.4	18.0	21.8	21.0
random2	5.5	6.1	12.2	7.3	9.0	23.5
random3	0.4	2.5	598.5	14.6	18.0	23.5
random4	0.4	2.8	533.3	21.6	27.0	25.0
random5	0.5	3.3	520.8	118.3	144.5	22.1
Average			436.0			23.0

Table 4.3. CPU time of design optimization flow

Benchmarks	CPU time (s)			
	RAMPS	PD-only	SR-only	EVO
auto	181.5	238.2	322.9	1383.1
consumer	315.9	240.6	1738.6	600.7
office4x	301.6	311.3	516.5	5327.6
networking	285.1	319.5	616.2	1036.2
telecom	478.2	514.5	5449.8	690.3
random1	956.2	1037.5	996.7	4846.3
random2	315.6	300.0	467.6	2848.6
random3	773.0	911.4	1402.4	3712.6
random4	1332.9	561.7	438.5	2559.9
random5	1419.4	458.3	4808.7	934.2
Average	635.9	489.3	1675.8	2394.0

First, as indicated by the super-linear dependence of area on MTTF, reliability comes at some cost in area but this cost is initially small, per year improvement in MTTF. Note that, in Figure 4.4, area is plotted on a logarithmic scale. As shown in Table 4.1, improving the average system MTTF over all benchmarks by 40%, 85%, and 180% results in maximum area overheads of 0.0%, 5.0%, and 10.0%. RAMS is sometimes able to improve MTTF without area

overhead because the architectural optimization moves it uses indirectly result in new floorplans that are as compact, or more compact, than the previous floorplan. However, this is rarely the case and can be viewed as noise. The initial solutions are optimized for area; they tend to have high power densities and temperatures. As a result, the temperature dependent fault rate is high. Area-optimized solutions also have low resource redundancy, i.e., a single hardware fault will often cause system failure. In addition, vulnerable points, i.e., hot, non-redundant PEs, normally exist in these systems. Therefore, for area-optimized initial solutions, the system reliability can be improved at low area cost. RAMS introduces PEs with lower power densities and/or replaces non-redundant PEs with redundant ones, thereby optimizing thermal properties and allowing the system to continue operating despite some runtime hardware faults.

Second, RAMS automatically trades off system reliability for area overhead. As shown in Table 4.1, it consistently produces a set of solutions with different area and reliability trade-offs, allowing system designers to choose a desirable solution based on domain-specific design constraints.

Third, as system MTTF increases, the area penalty associated with further improving system reliability increases, i.e., the areas of these application-specific MPSoCs are superlinearly dependent on MTTF. As shown in Table 4.1 and Table 4.2, RAMS achieves a significant 436% average system MTTF improvement with a maximum area overhead of 25%. Further improvements to system MTTF become prohibitively costly. This can be explained in the following way. PE failure cumulative distribution functions are non-decreasing. For some large duration, there is a low probability that any PE will operate without a fault. As a result, at very large MTTFs, adding PEs or reinforcing a subset of existing PEs with redundant components has little impact on MTTF.

4.4.3. Evaluation of Proposed Optimization Moves

RAMS optimizes system reliability by controlling PE temperatures and improving system redundancy. To evaluate the effectiveness of the proposed optimization moves, we compare RAMS with two alternative moves described in Section 4.3.5: power density only (PD-only) and component redundancy only (CR-only) moves. PD-only optimizes system MTTF by minimizing chip power density, i.e., replacing high power density PEs with low power density ones, and/or introducing extra low power density PEs to balance the workload and reduce chip power density. CR-only optimizes system MTTF by introducing system resource redundancy, i.e., adding redundancy to PEs and/or introducing new, redundant, PEs.

Figure 4.5 shows the results produced by RAMS as well as the CR-only and PD-only optimization moves. RAMS almost always produces solutions with better quality, i.e., lower area overhead for a given MTTF. This is indicated by its line generally falling below, and to the right, of the lines for the other synthesis algorithms, i.e., RAMS produces MPSoC architectures with both superior area and system MTTF. Notice that, in some cases, PD-only or CR-only may also produce high-quality solutions. Even though PD-only does not consider the component redundancy of PEs, introducing redundant PEs in order to improve power density still improves system MTTF. Even though CR-only does not consider PEs power density, redundant PEs tend to have lower power density than non-redundant ones. In general, it is necessary to exploit both structural redundancy and power density to consistently produce high-quality solutions.

4.4.4. Evaluation of Proposed Optimization Flow

RAMS uses an optimization flow in which the MOCSYN PRSA stochastic optimization algorithm is first used to optimize area, followed by an iterative improvement algorithm that

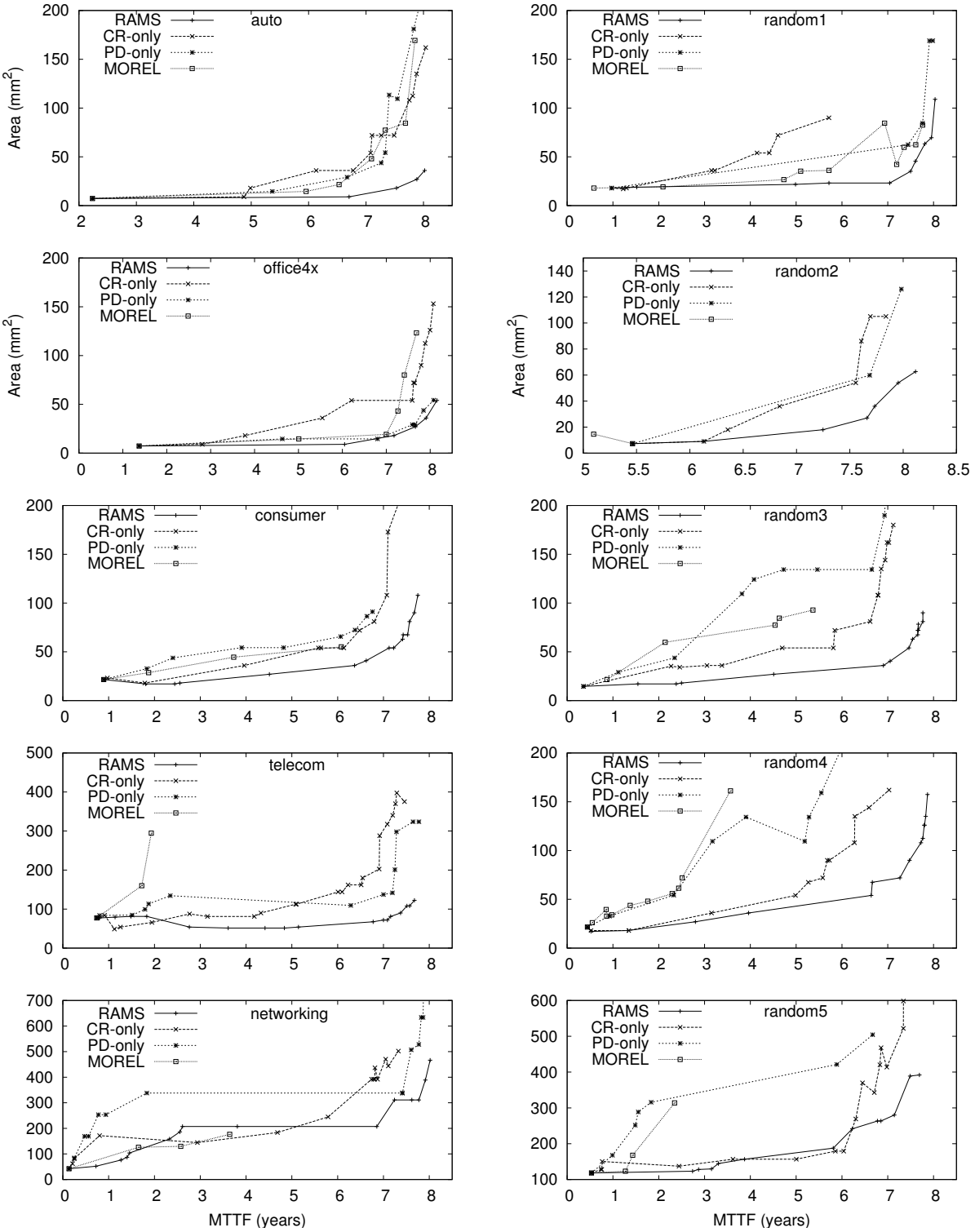


Figure 4.5. Comparison of different optimization heuristics.

uses domain-specific optimization moves to improve system MTTF while limiting area overhead. To determine whether a simpler reliable MPSoC optimization flow would be adequate, we compared RAMS with a version of MOCSYN^R that has been modified to take MTTF into account during stochastic optimization. MOCSYN^R does not use any MTTF-aware heuristics but does optimize MTTF by including it in its multi-objective cost function. We found that RAMS can almost always produce solutions with of equal or better quality than MOCSYN^R. Moreover, as shown in Table 4.3, RAMS generally produces better solutions in less CPU time than MOCSYN^R. In this table, CPU time includes the time for both optimization and solution analysis. We believe that the primary reason RAMS quickly produces solutions of better quality than MOCSYN^R is that the solution space of non-trivial reliable MPSoC synthesis problem instances is of such high dimensionality, and the cost of evaluating a solution so high, that unused tentative optimization moves consume so much CPU time that little time remains to advance toward more promising solutions. Recall that evaluating a solution requires solving multiple instances of floorplanning, network-on-chip synthesis, scheduling, and thermal analysis problems. Given enough time, MOCSYN^R may also be capable of finding high-quality solutions. However, given limited optimization time, the domain-specific knowledge built into RAMS generally allows higher-quality solutions to be produced.

4.5. Conclusion

This chapter has described a comprehensive method of synthesizing reliable application-specific MPSoCs. We have identified the key sub-problems within reliable MPSoC synthesis: (1) developing an architectural and physical design and analysis infrastructure that permits reliability calculation and optimization; (2) developing MPSoC component and system-level

reliability models, and efficient methods of computing MPSoC system MTTF; and (3) designing efficient domain-specific reliability optimization algorithms. These problems have been addressed with (1) RAMS, a comprehensive reliable MPSoC synthesis infrastructure that conducts architecture-level and physical-level synthesis and analysis, including floorplanning and thermal analysis, (2) unified component and system level reliability models incorporating electromigration, thermal cycling, TDDB, and stress migration, as well as efficient algorithms for computing MPSoC system MTTF, and (3) validated domain-specific reliability optimization algorithms that exploit redundancy and thermal profile optimization. Experimental results indicate that the resulting system is capable of improving MPSoC system MTTF by an average of 85% with less than 5% area cost and by an average of 436% with less than 25% area cost, compared to area-optimized solutions.

CHAPTER 5

Hybrid SET/CMOS Design for Low-Power Embedded Systems

Minimizing power consumption is vitally important in embedded system design; power consumption determines battery lifespan. Ultra-low-power designs may even permit embedded systems to operate without batteries by scavenging energy from the environment. Moreover, managing power dissipation is now a key factor in integrated circuit packaging and cooling. As a result, embedded system price, size, weight, and reliability are all strongly dependent on power dissipation.

Recent developments in nanoscale devices open new alternatives for low-power embedded system design. Among these, single-electron tunneling transistors (SETs) hold the promise of achieving the lowest power consumption. Unfortunately, most analysis of SETs has focused on single devices instead of architectures, making it difficult to determine whether they are appropriate for low-power embedded systems.

Evaluating the use of SETs in large-scale digital systems requires novel architectural and circuit design. SET-based design imposes numerous challenges resulting from low driving strength, relatively large static power consumption, and the presence of reliability problems resulting from random background charge effects. We propose a fault-tolerant, hybrid SET/CMOS, reconfigurable architecture, named IceFlex, that can be tailored to specific requirements and allows trade-offs among power consumption, performance requirements, operation temperature, fabrication cost, and reliability. Using IceFlex as a testbed, we characterize the benefits and limitations of SETs in embedded system designs. In particular, we focus on the use of SETs in

room-temperature ultra-low-power embedded systems such as wireless sensor network nodes. We also consider higher-performance applications such as multimedia consumer electronics. We see this work as a first step in determining the potential of ultra-low-power embedded system design using SETs. My major contribution of this chapter is on the global/local interconnect design and IceFlex microarchitecture characteristics.

5.1. Introduction

Energy consumption and thermal issues are now central in electronic system design. In high-performance applications, temperature affects integration density, performance, reliability, power consumption, and cost. For battery-powered embedded systems, energy consumption directly determines system life time. Power consumption crises were historically solved by moving to new technologies that decreased energy per operation, allowing increases in density and eventually performance. Power and thermal concerns were some of the main motivations for replacing vacuum tubes with semiconductor devices in the 1960s and replacing bipolar junction transistors with CMOS in the 1990s. Although CMOS is the mainstream fabrication technology used today, as IC and system integration further increase, it will reach fabrication, power consumption, and thermal limits; it may soon be time for another transition to a dramatically different technology.

Device researchers have seen the coming challenges for CMOS devices and evaluated alternative technologies such as carbon nanotube transistors [**115**], nanowires [**116**], and single-electron tunneling transistors (SETs) [**117**]. As projected by International Technology Roadmap for Semiconductors, SETs can potentially achieve the lowest projected energy per switching event of any known computation technology (1×10^{-18} J) [**2**]. However, their use poses

unique architectural, circuit design, and fabrication challenges. For example, SETs are susceptible to reliability problems resulting from $1/f$ noise caused by random background offset charges. They have cyclic I–V curves (see Figure 5.3) that can complicate design but permit highly-efficient implementation of some useful logic functions that have proven inefficient using CMOS and threshold logic. Although the fabrication of SETs capable of operating at low temperatures is now common, feature sizes of only a few nanometers are required for room-temperature operation. Fabricating SETs that function at room-temperature is challenging.

5.1.1. Past Work

After their discovery in the 1980s [**118**, **119**], there has been extensive research on fabrication, design, and modeling of SETs. Please refer to the survey by Likharev [**117**] for more details. SET fabrication and use in high-sensitivity amplifiers at cryogenic temperatures has been the main research focus in the past [**120**]. SETs and simple circuits with a variety of structures were proposed and fabricated using different methods and materials [**121**, **122**, **123**, **124**]. Recently, researchers have been able to fabricate SETs operating at room-temperature [**125**, **126**, **127**, **128**]. This work provides a promising start for SET circuit design. Various SET-based circuit applications, such as logic [**129**, **130**, **131**, **132**, **133**] and memory [**134**, **135**, **136**] have been developed. They demonstrate orders of magnitude improvement in power consumption and energy efficiency compared to CMOS.

Research into SET modeling and simulation has also been an active area. Monte Carlo simulation has been widely used to model SETs. SIMON [**137**] and MOSES [**138**] are the two most popular SET simulators. However, they are not suitable for circuit analysis due to large runtimes when characterizing systems containing more than a few SETs. Uchida et al.

proposed an analytical SET model and incorporated it into SPICE [**139**]. Recently, Inokawa et al. extended this model to a more general form to include asymmetric SETs [**140**]. Mahapatra et al. propose a simulation framework for hybrid SET/CMOS circuit design and analysis [**141**]. Their model for SET behavior is similar to that of Uchida et al.. These compact modeling techniques are efficient enough for use in SET circuit design and analysis and closely match Monte Carlo simulation results.

Significant challenges still remain for large-scale integration of SETs and for implementation in room temperature SET-based circuits. SETs that operate reliably at room temperature have critical dimensions of $\sim$1–10 nm. They are challenging to fabricate using current top-down lithographic techniques. However, several exciting advances make the evaluation of architectures for high density logic based on SETs worthwhile. Scanning-probe microscopes can be used to create devices smaller than those using conventional lithography [**125**]. Continual progress has been made on bottom-up nano-fabrication techniques, where chemical techniques are used to make individual molecules with useful electronic properties. Molecular quantum dots [**142**] can display SET behaviour in a single molecule. Larger structures, such as carbon nanotubes and nanowires can act as SETs [**124**]. These bottom-up techniques can create structures supporting room-temperature SET operation. However, more research is needed in order to integrate individual devices into large-scale circuits. Very recent advances in graphene [**143**] devices, in which a single atomic layer of graphite is used, show promise for SETs. In parallel with this work to make smaller devices, reliable methods for cooling to very low temperatures without supplies of liquid helium or nitrogen are becoming more common [**144**]. For high-performance computing, the added complexity of operating at cryogenic temperatures may not

be a limiting factor. Similarly, cryogenic temperatures are readily attained using passive methods in space.

5.1.2. Contributions

In this chapter, we explore the potential use of SETs in low-power embedded systems. In order to take advantage of the power efficiency of SETs, it is critical to bring SET-based design to the system level, characterize the impacts of SETs on system design metrics, and evaluate the benefits and limitations of SETs. Our work starts from design space characterization of SET-based architectures. We evaluate the impacts of using SETs upon architectural, circuit-level, and device-level design, considering metrics such as energy efficiency, performance, reliability, maximum operating temperature, and ease of fabrication.

Based on our evaluation of the architectural and circuit-level features that can most effectively exploit the strengths of SETs while working within the constraints their use imposes, we propose a fault-tolerant, reconfigurable, hybrid SET/CMOS based architecture called IceFlex. IceFlex is regular and cell-based, easing nanoscale design and fabrication. It is reconfigurable, permitting compensation for fabrication defects. It incorporates flexible, modular circuits to enable tolerance of run-time fault. In addition to compensating for the weaknesses of SETs, IceFlex exploits their strengths, e.g., we develop a two-SET design to implement Boolean functions that are not linearly separable, permitting fast and energy-efficient arithmetic.

We tailor IceFlex to both high-performance and battery-powered embedded systems and characterize its energy efficiency, performance, and power consumption using a number of instruction processors and application-specific cores. Compared to CMOS-based designs, IceFlex

improves energy efficiency by two orders of magnitude for both battery-powered and high-performance applications, while maintaining good performance. However, our results also indicate great challenges to the use of SET-based designs in portable embedded systems. Their use will either require advances in the compact cooling technologies or the fabrication of features with sizes approaching physical limits.

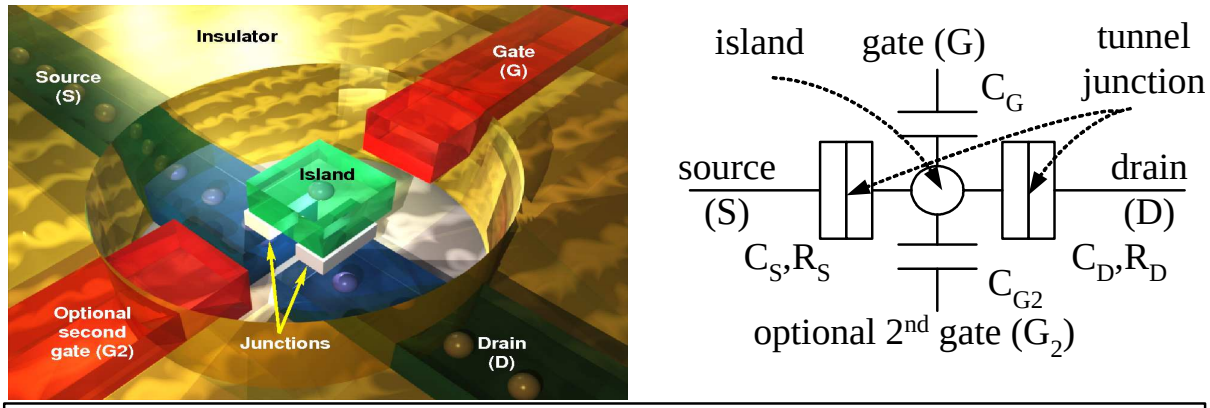
The rest of this chapter is organized as follows. Section 5.2 introduces SET operation and models. Section 5.3 describes IceFlex, the proposed hybrid SET/CMOS reconfigurable architecture. Circuit and architecture design and design tradeoffs will be discussed. Section 5.4 characterizes the IceFlex microarchitecture for use in embedded computing applications. We draw conclusions in Section 5.5.

5.2. SET Modeling

In this section, we introduce the physical properties of SETs, and discuss SET analytical device modeling.

5.2.1. SET Basics

The operation of a single-electron tunneling device is governed by the Coulomb charging effect. As shown in Figure 5.1, a single-electron tunneling device consists of a nanometer-scale conductive island embedded in an insulating material. Electrons travel between the island, source (S), and drain (D) through thin insulating tunnel junctions. When an electron tunnels into the island, the overall electrostatic potential of the island increases by e^2/C_Σ, where e is the elementary charge and C_Σ is the island capacitance. For large devices, this change in potential is negligible due to the high island capacitance C_Σ. However, for nanometer-scale



C_G :gate capacitance	C_D :drain tunnel junction capacitance
C_{G2} :optional 2nd gate capacitance	R_S :source tunnel junction resistance
C_S :source tunnel junction capacitance	R_D :drain tunnel junction resistance

Figure 5.1. SET structure and schematic.

islands, the capacitance C_Σ is much smaller. As a result the electrostatic energy change due to the addition or removal of a single electron can be larger than the thermal energy, particularly at low temperatures.

Changes to SET island potential results in an energy gap at the Fermi energy, preventing further electron tunneling. This phenomenon is called Coulomb blockade. It prevents current from flowing between source and drain ($I_{ds} = 0$), i.e., the SET is turned off. The Coulomb blockade effect can be overcome by changing the voltage of a conductor capacitively coupled to the island, thereby turning tunneling on and off. Although their transfer functions differ significantly from those of CMOS transistors, with careful circuit design, SETs can be used to realize logic functions either using circuits analogous to CMOS, or using radically different design techniques [**117**].

As shown in Figure 5.1, a SET typically has four terminals. The source and drain terminals (S, D) serve as electron reservoirs. When the SET is turned on, electrons tunnel from one terminal, through the junction, to the conductive island. They then tunnel through the other

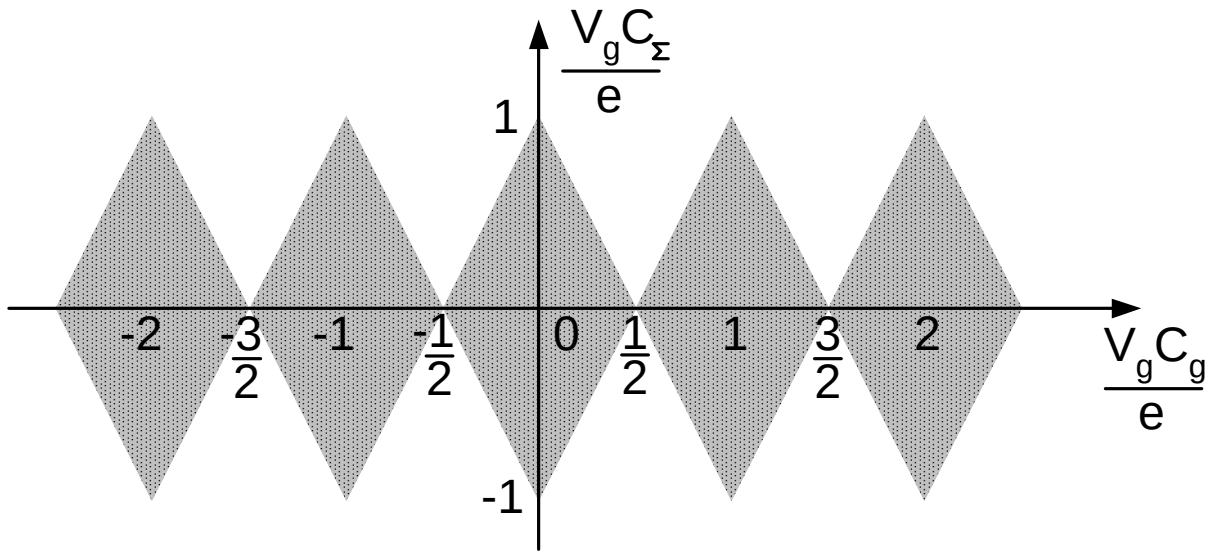


Figure 5.2. The Coulomb blockade effect of SET.

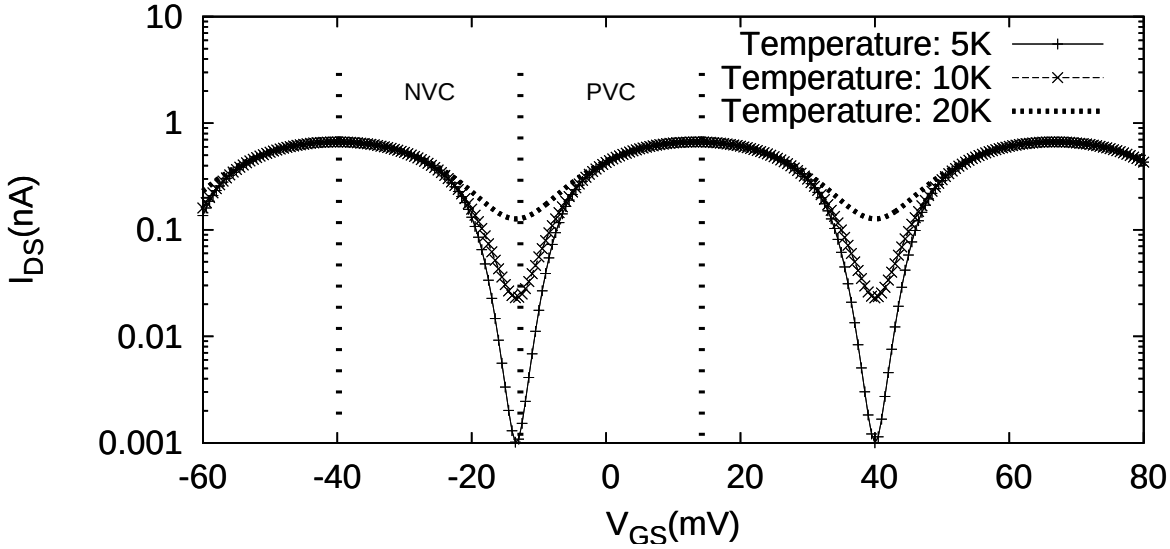


Figure 5.3. SET Coulomb oscillation (C_g =3.2 aF, $C_s = C_d$ =1.0 aF, and $R_s = R_d$ =10 MΩ).

junction to the other terminal. Each tunneling junction is modeled as a resistor (R_S or R_D) and a capacitor (C_S or C_D) in parallel. A gate terminal (G), with coupling capacitance C_G, controls

Table 5.1. Island size estimation. Assuming disc capacitor model ($C_\Sigma = 8\varepsilon r$). One side of the island is embedded into silicon dioxide. The other side of the island is exposed to Nitrogen.

Temperature (K)	$C_\Sigma = e^2/(10k_BT)$		$C_\Sigma = e^2/(40k_BT)$	
	Island capacitance (aF)	Island diameter (nm)	Island capacitance (aF)	Island diameter (nm)
40	4.65	52.48	1.16	13.12
77	2.41	27.26	0.60	6.82
103	1.80	20.38	0.45	5.10
120	1.55	17.49	0.39	4.37
200	0.93	10.50	0.23	2.62
250	0.74	8.40	0.19	2.10
300	0.62	7.00	0.15	1.75

the transport of electrons. A SET may also contain an optional second gate terminal (G_2), which is generally used to tune SET V_{GS} bias to the optimum operating point. The Coulomb blockade effect is maximized when $V_{GS} = me/C_G$, where $m = 0, \pm 1, \pm 2, \cdots$ [**145**] because, at these voltages, the system is in a minimal-energy state when an integer number of electrons are present on the island. Any single tunneling event between island and either source or drain would move the system from this state. The Coulomb blockade effect vanishes when $m = \pm 1/2, \pm 3/2, \cdots$, i.e., when m is a half integer value because, at these voltages, the system is in a minimal-energy state when a half-integer number of electrons are present on the island. In this case, a single tunneling event does not move the system from a minimum energy state. Electrons can therefore tunnel, in single-file, through the island as determined by V_{DS}.

Figure 5.2 illustrates the Coulomb blockade effect of a SET as a function of the gate voltage V_{GS}. The gray diamonds depict the regions in which the Coulomb blockade effect exists. This diagram demonstrates that the transfer functions of SETs differ dramatically from those of MOS devices. Discrete electron charging results in a periodic I–V transfer curve. The periodic changes are called Coulomb Oscillations. The I–V curve of a SET, shown in Figure 5.3, is

therefore periodic; drain current changes periodically as a function of the gate voltage, and has peaks and valleys with periods of e/C_g.

In order to observe the Coulomb blockade effect, the following constraints must be satisfied.

- Since thermal fluctuations can suppress the Coulomb Blockade effect, the electrostatic charging energy, e^2/C_Σ, must be much greater than k_BT, where k_B is Boltzmann's constant and T is the temperature. In order to ensure reliability, $e^2/C_\Sigma \geq 10k_BT$ or the more conservative $e^2/C_\Sigma \geq 40k_BT$ constraint is enforced. These equations imply that the maximum allowed island capacitance is inversely proportional to temperature. At room temperature, an island capacitance below 1 aF is required. Island capacitance is a function of island size. As shown in Table 5.1, using disc capacitor model, room-temperature operation requires an island size in the nanometer range, making fabrication challenging. At present, the smallest island capacitance of a fabricated device is around 0.15 aF [**127**].
- To observe single-electron charging effects, electrons must be confined in the island, which requires that the junction resistance be higher than the quantum resistance, i.e., $R_S, R_D > h/e^2$, $h/e^2 = 25.8\,\mathrm{k}\Omega$, where h is Planck's constant. Therefore, SETs have high resistances and low driving currents.

In order to operate voltage-state logic, SETs must exhibit voltage gain. The low-temperature voltage gain is equal to the gate capacitance divided by the sum of the junction capacitances: $G = C_G/(C_S + C_D)$. Achieving this gain requires techniques to reduce tunneling junction capacitance. It also requires close coupling of gate and island without a large increase in the

total island capacitance. High gain has only been demonstrated for a few devices and has required operation at low temperatures [**146**, **147**]. However further advances in lithography or nanofabrication may overcome this limitation.

5.2.2. Random Background Charge Effects

Background charge has been a persistent problem for SETs. Charges in the environment surrounding the SET island influence its equilibrium state. Defects near the island serve as charge traps [**148**]. Although the voltage offsets caused by these charges can be compensated for using a biased second gate terminal, the required bias cannot be known until after fabrication. Worse yet, some devices are affected by random background charge effects.

It is the tentative consensus of the research community that random background charge effects are caused by multiple, closely-spaced charge traps near the island, among which charge carriers may tunnel. Random background charge effects produce run-time variations in gate bias, and may cause logic errors. Much work has been done to understand the nature and density of these defects [**149**, **150**, **151**]. Most SETs have been fabricated with aluminum islands. Some researchers have attempted to eliminate the random background charge effects by designing SETs using alternative island materials such as silicon, based on the thesis that the use of non-crystalline, non-stoichiometric aluminum oxide junctions in conventional SETs leads to numerous charge-trapping defects. Silicon island SETs have shown greatly improved immunity to random background charge noise, with operation unchanged over several weeks [**152**]. However, random background charge effects remain the main source of run-time reliability problems for most SET designs. In this work, we describe a reconfigurable architecture that provides architectural resistance to the effects of random background charges.

5.2.3. SET Modeling

Circuit and architecture design involves extensive large-scale circuit simulation. Despite their accuracy, Monte Carlo methods are not suitable for large-scale circuit analysis due to their high time complexities. We build upon the SET analytical model developed by Inokawa et al. [**140**]. This compact model can be incorporated into SPICE. Combined with MOS transistor models, it provides an efficient and accurate simulation solution for hybrid SET/CMOS circuits. Inokawa's model ignores random background charge effects. In addition, it does not consider multi-gate effect, a necessary feature for modeling multi-gate SET circuit structure. In this work, we incorporate these two effects into Inokawa's model, and use it for hybrid SET/CMOS circuit and architecture design.

The I–V characteristics of a SET with island charge equal to n or $n+1$ electrons follow: where

$$C_{\Sigma} = C_S + C_D + \sum C_{G_i} \tag{5.1}$$

In this model, $\frac{2\sum C_{G_i} V_{GS_i}}{e}$ models the Coulomb charging effects of the multiple gate terminals. ζ is a real number that characterizes the random background charge effect.

This compact model is derived based on the steady-state master equation, which is not directly applicable to transient circuit analysis. However, when used in circuits, SETs are connected by metal wires. Based on existing fabrication processes, the capacitance of local interconnect is at least two orders of magnitude higher than SET island capacitance, thereby eliminating inter-SET Coulomb interaction. The independence of SETs enables the use of quasi-steady-state analysis [**140**, **153**].

5.3. IceFlex: A Fault-Tolerant Hybrid SET/CMOS Reconfigurable Architecture

This section describes the design and analysis of IceFlex, the proposed low-power, fault-tolerant, reconfigurable, hybrid SET/CMOS architecture. The vast majority of devices in IceFlex are SETs, allowing extremely low power consumption. CMOS devices are sparingly used to improve the driving strength of global interconnect.

Our evaluation of the architectural constraints imposed by SETs led to four main conclusions: (1) Flawless fabrication will be challenging, especially for circuits that operate at room temperature. It is important to simplify fabrication and use post-fabrication adaptation to avoid flawed devices; (2) An unpredictable subset of devices will be susceptible to random background offset charge effect noise: SET-based architectures should have the ability to tolerate run-time errors; (3)SETs have poor driving strength; this must be remedied, especially when driving global interconnect; (4)SETs have the ability to efficiently implement some functions that are inefficient using BJTs, CMOS logic, or threshold logic, e.g., non-linearly-separable functions can be implemented with one or two transistors: SET-based architectures should exploit such special properties in order to improve the efficiency of arithmetic and other logic circuits.

5.3.1. SET Design Space Characterization

In order to characterize the benefits and limitations of SET circuits and architectures, we analyze the tradeoffs among the following metrics: temperature, performance, power consumption, reliability, and fabrication constraints. This study yields two design configurations, each of which is shown in Table 5.2. One targets high-performance embedded applications such as

multimedia consumer electronics and one targets ultra-low-power battery-powered embedded applications.

5.3.1.1. Temperature. As shown in Table 5.2, IceFlex was evaluated at seven temperatures. IceFlex is a hybrid SET/CMOS design; the temperature range starts at 40 K to permit reliable operation of the CMOS components. 77 K is achieved by liquid nitrogen cooling. 103 K is the average cloud top temperature. 120 K and below are defined to be cryogenic. At 200 K, functional SET devices have been widely demonstrated in the literature. 250 K is a temperature that might be reached using a stacked Peltier heat pump. 300 K is room temperature.

5.3.1.2. Capacitance. To observe well-defined Coulomb charging effects, electron charging energy must be higher than the thermal energy that might be imparted to charge carriers, i.e., $\frac{e^2}{C_\Sigma} \geq 10k_BT$ or $\frac{e^2}{C_\Sigma} \geq 40k_BT$ to enable more reliable SET circuit design, where k_B is Boltzmann's constant and T is the temperature. At room temperature, this constraint requires an island capacitance below 1 aF, making fabrication challenging but possible [**127**]. In order to operate voltage-state logic, SETs must exhibit voltage gain, which is equal to the gate capacitance divided by the sum of the junction capacitances: $G = C_G/(C_S + C_D)$. Our results indicate that a gain of 1.5 is sufficient for use in digital logic. Targeting battery-powered systems, using $C_\Sigma \leq e^2/(10k_BT)$, $C_\Sigma \leq e^2/(40k_BT)$ and $G = 1.5$, the maximum allowed gate and junction capacitances are derived and shown in the "Low power, Capacitance" columns of Table 5.2.

SET performance degrades as device capacitance increases. However, decreased capacitance makes fabrication challenging. We assume the capacitances at 300 K are the minimal allowed gate and junction capacitances. Given $\frac{e^2}{C_\Sigma} \geq 10k_BT$, for high-performance applications, these minimal gate and junction capacitances are used at all the temperature settings and shown in the corresponding "High Performance, Capacitance" columns of Table 5.2. On the

other hand, given $\frac{e^2}{C_\Sigma} \geq 40k_BT$, which requires very low SET capacitance at room temperature ($C_G = 0.09aF$) and makes fabrication very challenging. Due to fabrication concerns, for high-performance design, the capacitance and voltage are determined by the corresponding operation temperature, instead of room temperature.

5.3.1.3. Voltage. Consider a SET biased via a second gate, such that a V_{GS} of zero places it in the middle of the positive voltage coefficient (PVC) region in Figure 5.3. In this case, the maximum range of current values can be traversed by letting V_{GS} (i.e., Vin) vary in the range $[-e/(4C_G), e/(4C_G)]$. At all but the lowest temperatures, this range also provides near-optimal sensitivity to V_{GS}. Therefore, we use this range. Once the range of V_{GS} is known, a V_{SS} of $-e/(4C_G)$ and a V_{DD} of $e/(4C_G)$ naturally follow, shown in the "Voltage" columns of Table 5.2. Note that a bias voltage applied via a second gate can be used to shift the zero V_{GS} point from the PVC to negative voltage coefficient (NVC) region in Figure 5.3, permitting NMOS-like or PMOS-like behavior.

5.3.1.4. Junction Resistance. To observe single-electron charging effects, electrons must be confined in the island. This requires junction resistances that are much higher than the quantum resistance, i.e., $R_S, R_D \gg h/e^2$, $h/e^2 = 25.8\,\mathrm{k\Omega}$, where h is Planck's constant. Therefore, SETs have high resistances and low driving currents. In this work, we pick two resistance settings: $100\,\mathrm{K\Omega}$ for high-performance applications and $10\,\mathrm{M\Omega}$ for battery-powered systems, shown in the "Resist." columns of Table 5.2.

5.3.1.5. Reliability Implications. Researchers have pointed out the dangers posed by thermal noise as charging (state change) energy approaches thermal energy. We explicitly consider the effects of temperature on steady-state current during circuit analysis and design, its effects are reflected in our design decisions, and their impacts on power consumption and performance

Table 5.2. Design Space Characterization

Temp. (K)	$C_\Sigma = e^2/10k_BT$								$C_\Sigma = e^2/40k_BT$							
	Low power				High performance				Low power				High performance			
	C (aF)		V (mV)	R (MΩ)	C (aF)		V (mV)	R (kΩ)	C (aF)		V (mV)	R (MΩ)	C (aF)		V (mV)	R (kΩ)
	C_G	C_S C_D	V_{dd}, V_{in} $e/4C_G$	R_S R_D	C_G	C_S C_D	V_{dd}, V_{in} $e/4C_G$	R_S R_D	C_G	C_S C_D	V_{dd}, V_{in} $e/4C_G$	R_S R_D	C_G	C_S C_D	V_{dd}, V_{in} $e/4C_G$	R_S R_D
40	2.78	0.93	14.36	10	0.37	0.12	107.70	100	0.70	0.23	57.46	10	0.70	0.23	57.46	100
77	1.45	0.48	27.65	10	0.37	0.12	107.70	100	0.36	0.12	110.60	10	0.36	0.12	110.60	100
103	1.08	0.36	36.99	10	0.37	0.12	107.70	100	0.27	0.09	147.95	10	0.27	0.09	147.95	100
120	0.93	0.31	43.09	10	0.37	0.12	107.70	100	0.23	0.08	172.37	10	0.23	0.08	172.37	100
200	0.56	0.19	71.82	10	0.37	0.12	107.70	100	0.14	0.05	287.28	10	0.14	0.05	287.28	100
250	0.45	0.15	89.77	10	0.37	0.12	107.70	100	0.11	0.04	359.10	10	0.11	0.04	359.10	100
300	0.37	0.12	107.70	10	0.37	0.12	107.70	100	0.09	0.03	430.91	10	0.09	0.03	430.91	100

are considered. Our model does not explicitly model the effects of temperature-dependent shot noise [**139**]. Therefore, it is necessary to design circuits with charging energies that are substantially higher than the thermal energy. Designs with charging energies of both 10 and 40 times the thermal energy are evaluated in this chapter ($10k_BT$ or $40k_BT$). Researchers have reported device operation at each level but the $40k_BT$ requirement is more reliable. At charging energies over $10k_BT$, the model we use is accurate to within 4% of the time-dependent master equation [**154**, **139**].

Random background charge effects [**150**, **151**] are the main barrier to SET reliability. They are observed as $1/f$ noise on SET gate voltages, with some SETs susceptible and others immune. Several recent devices have shown improved immunity to this noise, as described in Section 5.2.2. Currently, the distribution of random background offset charges can only be determined after fabrication [**117**]. Susceptible SETs may suffer transient errors infrequently, e.g., only once per day. In this work, we use architectural techniques to reduce the probability of failure using an entirely SET-based design. SETs are used in parallel to exploit the lack of SET-to-SET correlation in random background offset charge effects.

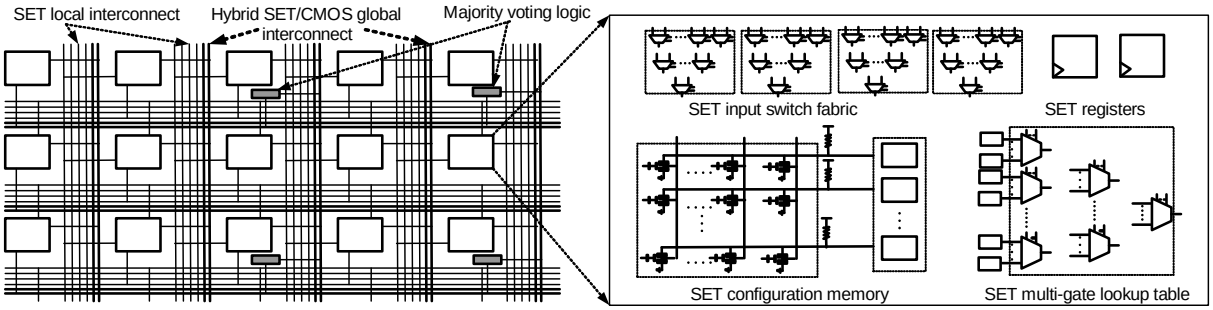


Figure 5.4. IceFlex microarchitecture

5.3.2. IceFlex Design

In this section, we present the architecture and circuit design of IceFlex. The microarchitecture of IceFlex is shown in Figure 5.4. IceFlex is a cell-based design. Each cell is a SET logic block (SELB) composed of the following components: (1) multi-gate SET-based reconfigurable look-up tables that can realize arbitrary n-input Boolean functions; (2) a SET-based arithmetic unit that allows efficient implementations of non-linearly separable arithmetic operations; (3) a SET-based reconfiguration memory array that caches multiple configuration contexts to support efficient run-time reconfiguration; (4) a multi-gate SET-based input switch fabric; and (5) SET registers. In addition, IceFlex also includes SET threshold logic-based majority voting logic units, allowing a flexible solution to run-time reliability problems. In IceFlex, a multi-level on-chip interconnect fabric forms inter-SELB connections. Local connections rely on a custom-designed, SET-driven, variable-length, constant-latency interconnect. This interconnect structure reduces power consumption and simplifies physical-level design automation problems, e.g., placement and routing. SETs have limited driving strength. Therefore, IceFlex uses hybrid SET/CMOS interconnect circuits to drive global interconnects.

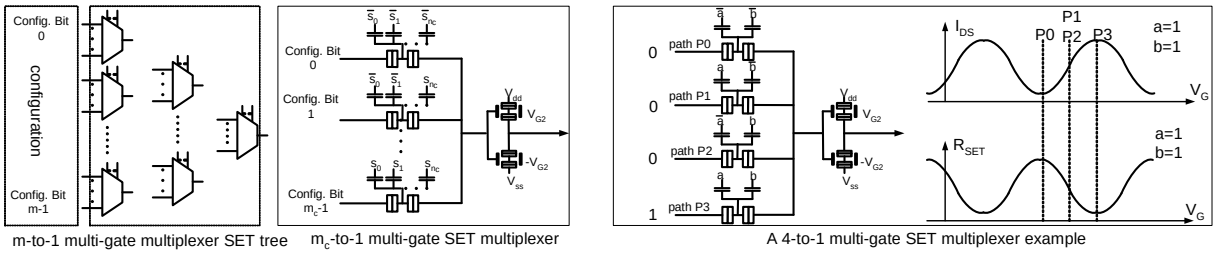


Figure 5.5. Multi-gate SET multiplexer tree.

We now explain the design of each IceFlex component and discuss the tradeoffs in circuit-level and architecture-level design.

5.3.2.1. Multi-Gate SET Reconfigurable Lookup Table Component. Each SELB is equipped with l sets of n-input reconfigurable look-up tables. Each look-up table can realize an arbitrary n-input Boolean function. The basic structure of the look-up table consists of an m-to-1 multi-gate SET multiplexer tree ($m = 2^n$), and an m-bit SET storage cell, which will be described in the next section.

The proposed multi-gate SET multiplexer tree differs from existing CMOS-based designs in the following way. A CMOS m-to-1 multiplexer tree requires $\lceil \log_2 m \rceil$ stages of transmission gates, plus buffers to meet the required driving strength. SETs may have multiple gate terminals. As described in Equation 5.1, the overall gate charging effect is a function of $\sum C_{G_i} V_{GS_i}$. Therefore, multiple control signals, e.g., the select signals for a multiplexer, can be supplied into a single SET, enabling a more compact circuit structure with better performance and power efficiency.

Figure 5.5 shows the proposed SET multi-gate multiplexer tree design. The basic building block is a q-to-1 multi-gate single-stage multiplexer, in which each of the q paths consists of a single multi-gate SET controlled by $\lceil \log_2 q \rceil$ select signals. Using this design, the logic depth of a n-to-1 multiplexer tree reduces to $\lceil \log_q m \rceil$ instead of $\lceil \log_2 m \rceil$. Figure 5.5 also shows a

design case for $q = 4$. The output SET buffer is used to break long resistive path and improve the driving strength.

As described in Section 5.2, thermal energy has significant impact on electron tunneling and the ratio of on to off currents, i.e., the ratio of the off to on resistance. This ratio decreases as the ratio of Coulomb charging energy (e^2/C) to thermal energy (k_BT) decreases. On the other hand, as the number of gate control signals per SET (hence the number of off paths connected in parallel) increases, the impact of the off paths on the circuit output increases. Consider, for the sake of example, the dual-gate 4-to-1 multiplexer design shown in Figure 5.5. The four logic inputs are 0001 and both select signals are logic one, i.e., $V_a = V_b = V$. Assume $C_a = C_b = C$. As shown in the I-V curve on the right side of Figure 5.5, for the SET on path $P3$, the overall gate charge equals $2CV$. Therefore, the SET becomes fully conductive. For paths $P1$ and $P2$, the gate charges both equal $CV - CV = 0$, hence both switches are partially conductive. For path $P0$, even though the overall gate charge equals $-2CV$, at high temperature its resistance may still be within the same order of magnitude as that of path $P3$. Since the inputs of paths $P0$, $P1$ and $P3$ are all connected to logic zero (the worst-case scenario), these three parallel paths may pull the output voltage below zero, producing incorrect results.

In the high-performance setting, the same capacitance settings are used across the whole temperature range. Therefore, the ratio of Coulomb charging energy to thermal energy increases as the temperature decreases. Therefore, lower temperatures permit fewer multiplexer levels in the multiplexer tree, with more inputs to each individual multiplexer.

Detailed circuit analysis shows that, using the high-performance setting and $e^2/C_\Sigma \geq 10k_BT$, the dual-gate design may be used at temperatures up to 200 K. At 250 K and 300 K, only the single-gate design is feasible. For the low-power setting, capacitance scaling maintains the

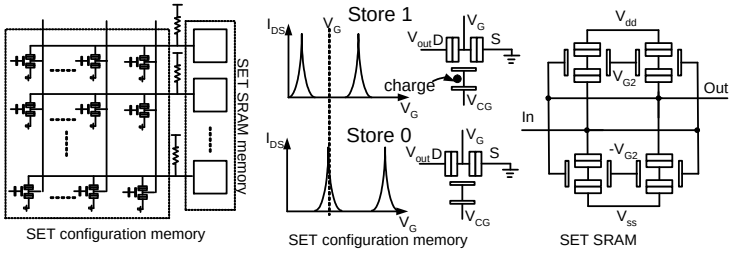


Figure 5.6. SET configuration memory.

same $e^2/C_{\Sigma}k_BT$ ratio. Therefore, the same design should be applied to the whole temperature range. In addition, since both the low-power setting and the high-performance setting at room temperature use the same $e^2/C_{\Sigma}k_BT$ ratio, only the single-gate design is feasible for low-power, room-temperature operation. For the $e^2/C_{\Sigma} \geq 40k_BT$ configurations of IceFlex, the dual-gate design may be used at all temperatures due to the increased charging energy.

5.3.2.2. SET Configuration Memory. In IceFlex, run-time reconfiguration is enabled by SET configuration memory, which consists of SET configuration cache and current configuration memory. In each SELB, the configuration cache stores multiple configurations. During run-time reconfiguration, one set of configuration bits stored in the configuration cache are placed into the current configuration memory to program SELB logic and interconnect. If k copies of configuration sets are stored in the configuration cache, then the circuit can be reconfigured k times during run-time execution without the need to access off-chip memory.

Figure 5.6 shows the circuit structure of the configuration memory in IceFlex. The SET configuration cache is the main on-chip configuration memory. Each storage cell consists of a dual-island SET [**117**]. A dual-island SET contains two capacitively-coupled SETs: a primary SET and a secondary SET. By controlling V_{CG}, electrons can tunnel through the control gate and charge the island of the secondary SET. The charge state of the secondary SET is able to shift the phase of the Coulomb oscillations of the primary gate, i.e., its conductivity condition

shifts as a function of gate control voltage, V_{GS}. Therefore, under a certain V_{GS}, the primary SET is either conductive or open due to different island charges, representing either a logic one or logic zero.

In the configuration cache, when a configuration is selected, a short-circuit path is formed between the pull-up resistor and SETs with a stored zero within the selected configuration set. The power consumption will be high if the configuration cache constantly controls the logic and interconnect. To minimize power consumption, IceFlex uses separate on-chip memories to store the currently-used configuration.

We designed two types of SET-based on-chip storage to hold the current configuration. The first design is a dual-island based SET buffer. As shown to the right of Figure 5.6, this buffer uses two opposite biasing voltages, V_{G_2} and $-V_{G_2}$, and behaves like a complementary SET inverter. During run-time reconfiguration, for each dual-island SET, the corresponding configuration bit stored in the configuration cache updates the island charge of its secondary SET, hence the conductivity of its primary SET, thereby controlling the buffer output. The second design is a SET SRAM design, which is similar to CMOS SRAM.

5.3.2.3. Efficient SET Implementations of Non-Unate Functions and Implications for Arithmetic. SETs have the ability to support efficient implementation of some critical logic functions that have long frustrated designers using threshold logic, BJT, and CMOS technologies. Most conventional transistors have either non-decreasing or non-increasing I–V curves. As a result, numerous devices are required to implement Boolean functions that are not unate, i.e., linearly separable. However, such functions are widely used, especially in digital arithmetic. The periodic nature of SET I–V curves can be exploited for efficient implementation of highly-useful non-unate functions such as exclusive-or.

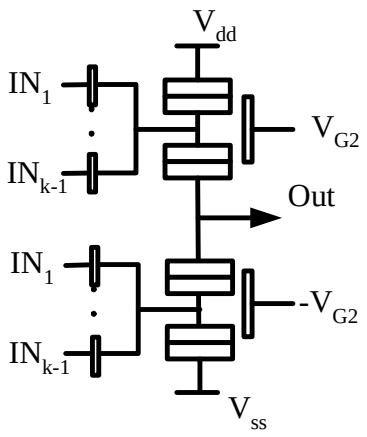


Figure 5.7. SET parity circuit.

The most efficient CMOS static pass-transistor logic design of a two-input exclusive-or gate in general use requires six transistors [**155**]. Moreover, it relies on strong input signals because it is not capable of signal restoration. A restoring version would require at least eight transistors. In contrast, it is possible to implement a two-transistor SET-based exclusive-or gate that is structurally equivalent to a CMOS inverter. In this design, each SET has two gates, each of which is connected to one of the exclusive-or inputs. The circuit structure for a SET-based *n*-input parity gate is shown in Figure 5.7. This design is capable of signal restoration. Thanks to the periodic SET I–V curve, it is possible to directly determine whether the number of high inputs is odd or even. By appropriately adjusting the gate capacitances, the device can be adjusted such that switching a single gate will result in a 180° phase shift in the I–V curve (see Figure 5.3). Note that even or odd parity functions with additional inputs may be implemented using only two SETs. The number of inputs is bounded primarily by geometrical constraints on fabrication of additional gates.

In SET-based architectures, we propose the use of fast carry chains based on the proposed exclusive-or (sum) computation logic. We have found that this design is approximately

75% more energy-efficient and 25% faster than a design based on a conventional CMOS-style exclusive-or sum implementation, when both are implemented using SETs. This design style is impossible for threshold logic, BJTs, and CMOS technologies. Note that carry-out logic is equivalent to 2-out-of-3 majority vote logic.

5.3.2.4. Reconfigurable Interconnect Network. IceFlex consists of a variety of reconfigurable interconnect resources, including SET local interconnects, hybrid SET/CMOS global interconnects, and SET switch fabric.

Interconnect consumes a substantial proportion of total power consumption in IceFlex: its power efficiency is critical. For SET-based interconnect, the static power consumption dominates due to the impact of thermal energy on device conductance, especially at high temperatures. In addition, static power consumption increases with wireload; maintaining unchanged communication latency with higher wireload requires lower junction resistance. In contrast, the dynamic power consumption of SETs is low due to the low SET gate capacitance and low voltage swing. For hybrid SET/CMOS-based interconnect, SETs are only used to drive CMOS buffers, which in turn drive metal wires. In this case, SETs with low driving strength, hence high junction resistance, are allowed. Compared to SETs, CMOS has lower static power consumption but higher capacitance and dynamic power consumption. Therefore, dynamic power dominates in the hybrid SET/CMOS-based design. Detailed circuit analysis shows that, under the same performance constraint, SET-based design is more energy-efficient for local interconnect and the hybrid SET/CMOS design is more energy-efficient for global interconnect.

In IceFlex, local interconnects driven directly by SET buffers support communication between nearby SELBs. Three types of local interconnects are supported: single length, double length, and hex length. The proposed SET local interconnect design guarantees a constant

Table 5.3. Impact of Majority Vote Logic on SELB Fault Probability

SET fault prob.	$1/1,000$			$1/10,000$			$1/100,000$		
Majority vote inputs	3	5	7	3	5	7	3	5	7
Raw fail prob.	6.20E-2	6.20E-2	6.20E-2	6.38E-3	6.38E-3	6.38E-3	6.40E-4	6.40E-4	6.40E-4
Best prob.	1.11E-2	2.17E-3	4.45E-4	1.22E-4	2.57E-6	5.71E-8	1.23E-6	2.62E-9	5.86E-12
SET MVL prob.	1.11E-2	2.18E-3	4.57E-4	1.22E-4	2.69E-6	1.77E-7	1.23E-6	3.82E-9	1.21E-9

latency across different routing lengths. Consider, for the sake of example, a local communication architecture in which the maximum interconnect delay is bounded by some value and the longest interconnect is appropriately buffered to meet this constraint. In this case, it would possible to similarly drive shorter interconnects, thereby decreasing their delays, relative to that of the longest interconnect. It would also be possible to reduce the driving strength on shorter interconnects to reduce power consumption and produce a local interconnect architecture in which all interconnects have uniform delay. We propose the second design because it improves interconnect power efficiency and also simplifies placement and routing during physical design.

The proposed SET local interconnect is designed as follows. A SET buffer with minimal driving strength (hence high junction resistance) is first determined. Next, for local interconnects with different routing lengths, minimal driving strength SET buffers are connected in parallel to meet driving strength requirements imposed by performance constraints. The main motivation of using parallel SET buffers is that SET junction resistance cannot be reduced arbitrarily ($R_D, R_S \gg h/e^2$). In addition, using homogeneous SET buffers in parallel instead of heterogeneous SET buffers is likely to simplify analysis and fabrication.

Remote connections introduce the high capacitive loads of long metal wires. To address the driving strength problem of SET-only circuits, we have designed hybrid SET/CMOS interface circuitry to drive global interconnect. Figure 5.8 shows the circuit structure, which contains two complementary SET inverters and two CMOS inverters. SELB's output is first fed to the input

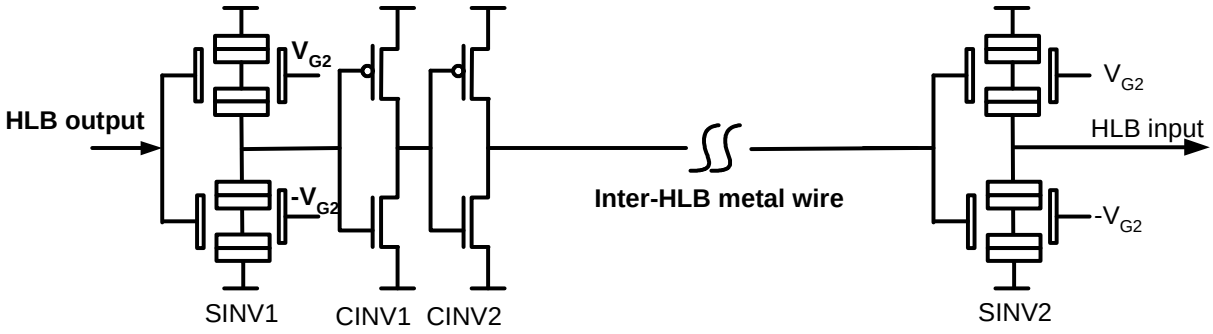


Figure 5.8. Hybrid SET/CMOS interface circuitry

of SET inverter SINV1. SINV1 drives the CMOS inverter, CINV1. Unlike the SET logic used inside SELBs, SINV1 uses a low-resistance design to improve driving strength. Fortunately, it is possible to achieve sufficient driving strength with a single SET so using parallel SETs is unnecessary. Since the voltage range of SET logic is much smaller than that of CMOS logic, the output signal of SINV1 is within the switching range of the CMOS inverter. Since both MOS transistors are conductive within the switching region, short-circuit power is high. To solve the short-circuit power consumption problem, CINV1 is designed to satisfy the following two constraints. First, $V_{tn} + |V_{tp}| > V_{dd} - V_{ss}$ ensures that at least one MOS transistor is off at all times, reducing static power consumption. Second, the output signal range of SINV1 must be greater than $V_{tn} + |V_{tp}| - (V_{dd} - V_{ss})$. Therefore, the NMOS (PMOS) transistor of CINV1 is conductive at output signal high (low) of SINV1 to provide enough driving strength to CINV2. CINV2 is a normal CMOS inverter. Therefore, CINV1 serves as a signal amplifier, and CINV2 provides driving strength.

CINV2 cannot be used to drive the input SET logic of a SELB directly. SET current is a periodic function of the gate control voltage and has a period of e/C_G. The output voltage range of CINV2 is much larger than e/C_G. Therefore, using this output voltage range directly

would cause output signal oscillation in SET logic. To solve this problem, we design a special SET inverter, SINV2, that is used for SELB inputs. SINV2 is fabricated with a large distance between gate and island in order to reduce the gate capacitance, C_G. Thus, e/C_G can match the output signal range of CMOS inverter CINT2. Recall that, although source–island and drain–island junctions must be short to permit tunneling, there is no such bound on gate–island separation.

In IceFlex, each SELB is equipped with a reconfigurable input switch fabric that selects the connections among local and global interconnects. The input switch fabric is implemented using multi-gate SET multiplexor tree, the same as the reconfigurable look-up table described in Section 5.3.2.1.

5.3.2.5. Design and Modeling of IceFlex Majority Voting Logic. Although researchers are making progress on reducing the severity of noise resulting from random background offset charge effects, it may continue to pose run-time noise problems in the future. Even if this problem can be entirely solved, resistance to run-time faults may be useful in SETs, e.g., to allow resistance to Alpha particle induced faults or other single event upsets. IceFlex incorporates support for hierarchical spatial redundancy to improve fault tolerance. Although much of the literature predicts the need for fault-tolerant architectures in nanoelectronics, the level of fault tolerance is currently unknown. Therefore, we consider the results for a number of possible SET failure rates and in the presence of three fault-tolerance configurations.

Other researchers have proposed a number of architectural techniques to support reliable computation using nanoscale electronics that are susceptible to fabrication-time and run-time faults. Dehon described the use of structural redundancy and programming-time defect-aware

configuration in a carbon nanotube and silicon nanowire based programmable logic array architecture [**156**]. Goldstein et al. describe the use of a defect map that is generated during post-fabrication testing to avoid the use of faulty devices [**157**]. Bahar et al. present a method of expressing logic circuits using Markov Random Fields, permitting Boolean functions to be computed using devices susceptible to potentially frequent transient faults [**158**]. We think it likely that the random background charge problem will ultimately be dealt with by a combination of improved fabrication technology, post-fabrication testing to identify and avoid a subset of the affected SETs, and run-time fault-tolerance via conventional structural redundancy or recent advances in probabilistic computation. IceFlex provides for regular structural redundancy and run-time error correction.

We now consider the fault model for IceFlex SELBs. Every path from SELB input to output contains 64 SETs. In the third row of Table 5.3, we show the SELB raw failure probabilities, i.e., the probability of a SELB producing an incorrect output. SELB failure probability is a function of the SET fault probability, for which Table 5.3 shows three values. Likharev estimates the long-term density of background offset charge susceptible SETs [**117**]. We follow his assumptions but correct a typographical error in that chapter, arriving at one susceptible SET in 10,000. The resulting $1/f$ noise produces long-duration failure periods. Therefore, in this analysis, we (conservatively) assume that susceptible devices consistently fail. In reality, errors may not be consistent, preventing post-fabrication identification of all susceptible devices. We also consider the higher SET fault probability of 1/1,000 and the lower fault probability of 1/100,000. Advances in fabrication and detection of most SETs susceptible to random background offset charge effects by post-fabrication testing may permit reduction in run-time SET fault probability.

We have considered the effect of using no MVL (Raw fail prob.), fault-free MVL (Best prob.), and SET MVL. Using a given reliability configuration, it is not possible for MVL-based designs to produce lower SELB fault probabilities than those shown in the Best prob. row. SET MVLs are constructed from multi-gate SETs. We focus on the three-input SET MVL design to simplify depiction; the five-input, and seven-input SET MVL follows an analogous design style. This circuit has identical structure to the parity gate shown in Figure 5.7. However, the separation of gates and island are adjusted such that the circuit traverses only 1/2 Coulomb oscillation period during use. The SET pull-up gates are separated sufficiently to require the majority of the gates to be high. The converse is true of the pull-down gates. For each SET depicted in Figure 5.7, four SETs are used in parallel in order to permit the failure of one SET while still producing correct results. We have computed the delay of the SET MVL by considering the worst-case scenario, in which one a path that is 3/5 or 4/7 closed has a faulty driver SET and the path that is 2/4 or 3/7 closed has no faulty SETs.

As shown in Table 5.3 it is possible for a seven-input SET-only MVL with redundant SELBs to reduce the failure rate to 1/8,500,000, given a SET fault probability of 1/10,000, or 1/830,000,000, given a SET fault probability of 1/100,000. Given recent trends in noise-resistant SET design and fabrication, it seems likely that a less aggressive fault tolerance configuration will be necessary in the future (see Section 5.2.3). However, when we later consider the impact of fault tolerance on energy efficiency and performance, we assume the use of seven-input SET MVLs for every SELB stage.

If a method of rapidly determining which SETs are susceptible to random background charge effects is ever developed, these effects can be avoided in the same way that fabrication defects are avoided: via the use of a regular computation structure in which operations are

mapped only to operational devices, which are determined post-fabrication. There has been some promising work on this topic, in which illumination is used to produce ions, accelerating the onset of random background charge effects [**159**].

5.4. Experimental Results

In this section, we evaluate the suitability of using SETs in low-power embedded system design. We start from the microarchitecture characterization of IceFlex. IceFlex is then used as a testbed to characterize the benefits and limitations of SETs for both high-performance and battery-powered embedded application.

5.4.1. Characterization of the IceFlex Architecture

Following the design parameters shown in Table 5.2, we evaluate the performance and power consumption of IceFlex using HSPICE. For SET circuitry, the SPICE model and device parameters are described in Section 5.2.3. For CMOS logic and metal wire, we use the 22 nm Berkeley BSIM4 predictive technology model, which models the impact of temperature on MOS devices. We analyze designs adhering to the $C_\Sigma = e^2/(10k_BT)$ constraint as well as the more conservative $C_\Sigma = e^2/(40k_BT)$ constraint. A low-power setting (targeting megahertz range) and a high-performance setting (targeting gigahertz range), are considered.

Tables 5.4, 5.5, 5.6, and 5.7 summarize the performance and power characterization of the logic components and interconnect fabric of IceFlex, including multi-gate SET reconfigurable lookup table (LUT)[1], SET register (Register), SET and CMOS four-out-of-seven majority voting logic (MVL), multi-gate (MG) and CMOS-style (CS) exclusive-or, (CO) carry-out logic, and SET local interconnect (Single, Double, and Hex), hybrid SET/CMOS global interconnect

[1]To allow comparison with Xilinx FPGAs, a 16-to-1 setting is used.

Table 5.4. Characterization of IceFlex Microarchitecture for $C_\Sigma = e^2/(10k_BT)$

				Low power							High performance						
				40 K	77 K	103 K	120 K	200 K	250 K	300 K	40 K	77 K	103 K	120 K	200 K	250 K	300 K
Latency (ns)	LUT			4.36	3.30	3.05	3.04	2.65	2.56	2.51	0.04	0.03	0.02	0.02	0.02	0.02	0.02
	Register			1.31	0.93	0.81	0.75	0.61	0.59	0.58	0.01	0.01	0.01	0.01	0.01	0.01	4.81e-3
	7-INPUT MVL			0.29	0.28	0.28	0.28	0.28	0.27	0.27	7.42e-4	1.59e-3	1.99e-3	2.18e-3	2.10e-3	1.91e-3	1.69e-3
	SET-MVL			0.76	0.44	0.49	0.49	0.48	0.48	0.48	0.01	0.01	0.01	0.01	0.01	0.01	4.87e-3
	Arithmetic Logic	SUM	MG	0.33	0.31	0.30	0.30	0.30	0.29	0.29	0.01	0.01	0.01	0.01	4.60e-3	3.74e-3	3.2e-3
			CS	0.60	0.53	0.51	0.51	0.49	0.48	0.47	0.01	0.01	0.01	0.01	0.01	0.01	0.01
		CO		0.76	0.44	0.49	0.49	0.48	0.48	0.48	0.01	0.01	0.01	0.01	0.01	0.01	4.87E-03
Power (nW)	LUT			0.08	0.31	0.55	0.75	2.09	3.26	4.70	3.41	19.29	35.37	46.84	96.12	343.78	441.44
	Register			0.04	0.15	0.26	0.35	0.98	1.53	2.20	3.75	18.38	31.83	38.37	66.36	174.39	206.66
	7 INPUT-MVL			0.02	0.08	0.14	0.19	0.54	0.84	1.21	1.97	14.41	26.46	34.75	72.91	94.37	113.50
	SET-MVL			3.44e-03	0.01	0.02	0.03	0.09	0.13	0.19	1.11	3.62	6.23	8.06	16.89	22.14	27.03
	Arithmetic Logic	SUM	MG	2.89e-03	0.01	0.02	0.03	0.07	0.11	0.16	0.23	0.42	0.92	1.50	6.55	10.79	15.29
			CS	4.89e-03	0.02	0.03	0.04	0.12	0.19	0.28	2.03	6.54	10.54	13.05	22.11	26.40	30.34
		CO		3.44e-03	0.01	0.02	0.03	0.09	0.13	0.19	1.11	3.62	6.23	8.06	16.89	22.14	27.03

Table 5.5. Characterization of IceFlex Microarchitecture for $C_\Sigma = e^2/(40k_BT)$

				Low power							High performance						
				40 K	77 K	103 K	120 K	200 K	250 K	300 K	40 K	77 K	103 K	120 K	200 K	250 K	300 K
Latency (ns)	LUT			10.04	7.86	7.09	6.80	5.57	5.03	4.75	0.08	0.06	0.05	0.05	0.05	0.04	0.04
	Register			1.42	1.09	1.02	1.00	0.90	0.88	0.86	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	7-INPUT MVL			0.58	0.57	0.58	0.58	0.59	0.56	0.58	3.28E-03	3.18E-03	3.16E-03	3.20E-03	3.24E-03	2.99E-03	3.14E-03
	SET-MVL			1.15	1.13	1.13	1.00	1.08	1.04	1.06	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	Arithmetic Logic	SUM	MG	2.32	2.31	2.31	2.31	2.31	2.28	2.29	0.01	0.01	0.01	0.01	0.01	0.01	0.01
			CS	3.02	2.97	2.95	2.96	2.95	2.89	2.93	0.01	0.01	0.01	0.01	0.01	0.01	0.01
		CO		1.15	1.13	1.13	1.00	1.08	1.04	1.06	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Power (nW)	LUT			0.07	0.26	0.44	0.58	1.60	2.64	3.70	6.67	25.76	44.53	58.19	162.20	266.69	373.81
	Register			0.08	0.30	0.53	0.72	1.99	3.14	4.48	8.02	29.88	53.16	72.12	199.64	315.21	450.34
	7 INPUT-MVL			0.05	0.20	0.36	0.48	1.32	2.17	3.02	5.37	20.05	35.87	48.15	132.24	217.31	302.60
	SET-MVL			0.01	0.03	0.06	0.08	0.21	0.34	0.48	0.94	3.51	6.26	8.44	23.24	37.58	52.90
	Arithmetic Logic	SUM	MG	1.61E-03	0.01	0.01	0.01	0.04	0.07	0.09	0.22	0.80	1.44	1.91	5.19	8.88	12.04
			CS	0.01	0.04	0.07	0.09	0.25	0.40	0.57	1.04	3.87	6.90	9.30	25.60	41.51	58.35
		CO		0.01	0.03	0.06	0.08	0.21	0.34	0.48	0.94	3.51	6.26	8.44	23.24	37.58	52.90

Table 5.6. Characterization of IceFlex Interconnect Fabric For $C_\Sigma = e^2/(10k_BT)$

		Low power							High performance						
		40 K	77 K	103 K	120 K	200 K	250 K	300 K	40 K	77 K	103 K	120 K	200 K	250 K	300 K
Latency (ns)	ISF	3.487	2.642	2.439	2.430	2.120	2.047	2.006	0.024	0.019	0.016	0.015	0.012	0.019	0.018
	Single	0.456	0.411	0.368	0.359	0.340	0.338	0.388	0.006	0.006	0.005	0.005	0.004	0.004	0.003
	Double	0.375	0.353	0.346	0.342	0.330	0.330	0.381	0.006	0.006	0.005	0.005	0.004	0.004	0.003
	Hex	0.164	0.340	0.336	0.333	0.325	0.326	0.377	0.006	0.006	0.005	0.005	0.004	0.004	0.003
	Global	1.651	4.214	7.612	7.505	5.083	5.502	7.968	0.169	0.157	0.143	0.139	0.123	0.123	0.131
Power (nW)	ISF	0.313	1.163	2.071	2.823	7.848	12.243	17.642	10.712	63.684	117.156	155.302	318.632	1292.302	1658.224
	Single	0.004	0.013	0.023	0.032	0.090	0.140	0.191	1.394	4.494	6.903	8.394	14.364	17.634	20.242
	Double	0.007	0.026	0.046	0.063	0.179	0.279	0.383	2.788	8.988	13.805	16.788	28.728	35.267	40.484
	Hex	0.014	0.052	0.092	0.126	0.358	0.558	0.766	5.576	17.977	27.611	33.575	57.455	70.535	80.968
	Global	383.830	53.908	19.761	18.014	14.309	12.807	10.903	3105.300	3122.000	3280.700	3249.900	3733.700	5042.500	9828.700

Table 5.7. Characterization of IceFlex Interconnect Fabric For $C_\Sigma = e^2/(40k_BT)$

		Low power							High performance						
		40 K	77 K	103 K	120 K	200 K	250 K	300 K	40 K	77 K	103 K	120 K	200 K	250 K	300 K
Latency (ns)	ISF	6.696	5.238	4.727	4.537	3.712	3.351	3.169	0.050	0.039	0.037	0.036	0.030	0.028	0.027
	Single	0.728	0.699	0.694	0.697	0.799	0.770	0.784	0.006	0.006	0.006	0.007	0.005	0.005	0.005
	Double	0.704	0.687	0.685	0.689	0.794	0.766	0.781	0.006	0.006	0.006	0.007	0.005	0.005	0.005
	Hex	0.692	0.680	0.680	0.684	0.791	0.763	0.779	0.006	0.006	0.006	0.007	0.005	0.005	0.005
	Global	2.996	4.523	4.657	4.237	4.572	4.520	6.785	0.163	0.110	0.092	0.086	0.074	0.073	0.099
Power (nW)	ISF	0.219	0.844	1.457	1.903	5.302	8.727	12.226	22.022	85.034	146.920	191.957	535.072	879.837	1233.147
	Single	0.008	0.032	0.057	0.076	0.210	0.342	0.479	0.959	3.387	6.193	7.977	24.992	34.101	53.581
	Double	0.017	0.063	0.113	0.152	0.420	0.684	0.958	1.917	6.775	12.386	15.955	49.984	68.202	107.160
	Hex	0.034	0.127	0.226	0.305	0.840	1.368	1.917	3.835	13.549	24.771	31.909	99.967	136.400	214.320
	Global	271.780	23.912	6.668	4.460	3.555	4.513	5.857	6674.800	5146.700	5560.900	5824.100	5318.200	4856.100	4745.700

(Global) and SET input switch fabric (ISF). From these results, we draw the following observations.

First, IceFlex has high energy efficiency, good performance, and high flexibility in terms of performance and energy efficiency tradeoff. At the low-power setting, the power consumptions of SET-based logic components and local interconnect fabric are within the range of nano-Watts. The hybrid SET/CMOS global interconnect has the highest power consumption. This is a result of the high capacitance of global wires and high power consumption of the CMOS buffers. For performance, all components in the low-power version of IceFlex still have latencies in the range of nanoseconds. SETs have high junction resistance and low driving strength. Using the high-performance setting, by scaling the SET junction resistance down to 100 kΩ, the latencies of the SET-based logic and local interconnect fabric are consistently lower than 100 ps. Even though reducing resistance results in a 100× increase in power, as demonstrated in Section 5.4.2, the overall energy efficiency of IceFlex is still orders of magnitude higher than that of CMOS-based solutions.

Second, these results demonstrate the impact of temperature on SET performance and power consumption – as the temperature increases, performance increases and the power efficiency decreases. This is a result of the impact of thermal energy on tunneling events and therefore circuit behavior, which is described in Section 5.2. The number of electrons with sufficient energy to overcome the Coulomb blockade effect increases with temperature, thereby increasing electron tunneling and therefore average device current. At high temperature, the average device conductance increases, increasing both performance and power consumption.

The $C_\Sigma = e^2/(40k_BT)$ setting enables greater resistance to shot noise than the $e^2/(10k_BT)$ setting. However, it also imposes performance and power consumption penalties. For SET circuitry, the required supply voltage is inversely proportional to SET gate capacitance. Compared to the $C_\Sigma = e^2/(10k_BT)$ setting, $C_\Sigma = e^2/(40k_BT)$ requires a further reduction of SET gate capacitance and an increase in supply voltage. Note that the driven capacitance of SET circuit is dominated by the metal wires. Therefore, decreased gate capacitance has negligible impact on power consumption. The increased supply voltage, on the other hand, increases circuit dynamic power consumption. Moreover, the increased voltage range increases the duration of signal swing, thereby increases circuit latency.

5.4.1.1. SET Multi-Gate Multiplexer Tree. As described in Section 5.3.2.1, multi-gate SETs benefit the performance, power consumption, and area efficiency of the multiplexer tree design. This section characterizes the impact of thermal energy on the proposed multi-gate design.

As described in Section 5.3.2.1, at the high-performance $C_\Sigma = e^2/(10k_BT)$ setting, the dual-gate design is used for temperatures at or below 200 K. For these settings only single-gate design is feasible at temperatures greater than 250 K due to high static current at these temperatures. As a result, circuit is decreased at high temperatures. As shown in Figure 5.9, from 200 K to 250 K, we observe both an increase in latency and power consumption. In addition, when using the same design, we observe that both the circuit performance and power consumption increase with temperature. The same trend was described in Section 5.4.1. Using the low-power design of IceFlex, only the single-gate design is feasible (see Section 5.3.2.1). The results are summarized in Table 5.4. Using $e^2/C_\Sigma \geq 40k_BT$, SET circuitry is less susceptible to thermal energy thanks to the increased charging energy. Therefore, both low-power and high-performance dual-gate multiplexer tree designs become feasible across the entire temperature range. As shown in

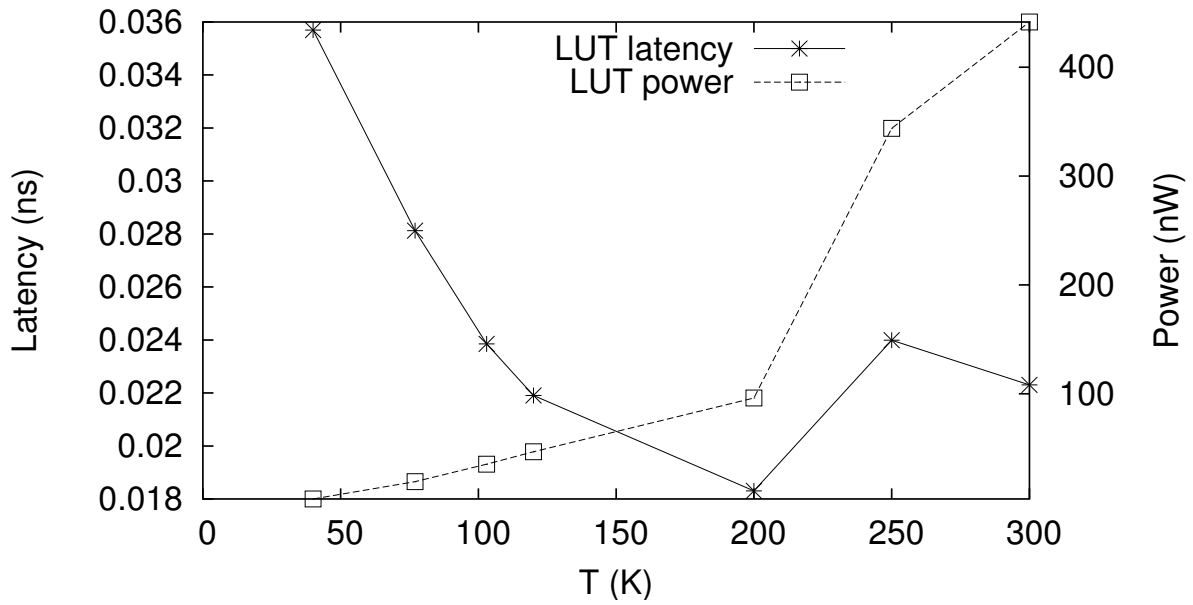


Figure 5.9. Power and performance of the multi-gate SET multiplexer tree for high performance, $C_\Sigma = e^2/(10k_BT)$

Figure 5.10, using the high-performance $C_\Sigma = e^2/(40k_BT)$ setting, the performance and power consumption of the multi-gate multiplexer tree design increase consistently with temperature. Similar trend can be shown for the corresponding low-power design case.

5.4.1.2. Power and Performance of Interconnect Design. Power consumption, performance, and the tradeoff between them are of central importance in interconnect design. We considered both SET-only and SET/CMOS hybrid interconnect driver designs. The power consumption ratios shown in Figures 5.11, 5.12, 5.13, and 5.14 characterize the power consumption of the SET-only and the SET/CMOS hybrid interconnect as a function of the wireload at 300 K at both the high-performance and the low-power settings. They show the ratios of power consumptions for the SET-only and SET/CMOS design power consumptions at the same performance. The figure indicates that the relative static power benefit of the SET/CMOS hybrid design over the SET-only design increases as the wireload increases. This is mainly due to an increase in the

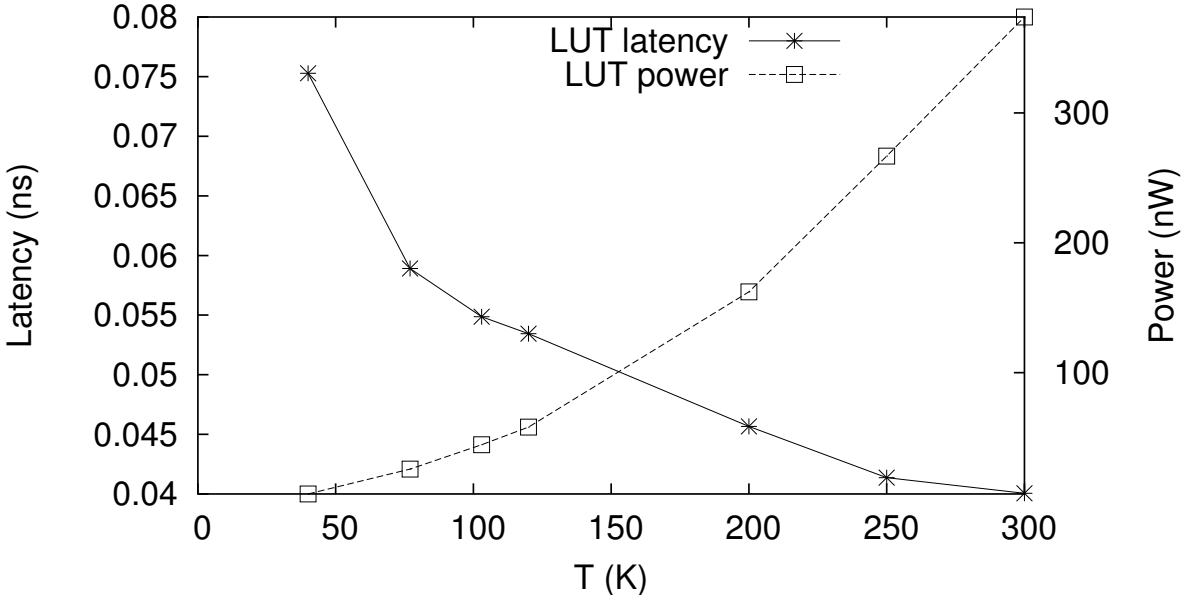


Figure 5.10. Power and performance of the multi-gate SET multiplexer tree for high performance, $C_\Sigma = e^2/(40k_BT)$

static power consumption of the SET-only design as more SET buffers are used to meet the driving strength requirements. The SET-only design has superior power efficiency. As the wire length increases, the proportion of capacitance contributed by CMOS buffer gates becomes less significant relative to wire capacitance. Therefore, compared to the SET-only design, the dynamic power consumption of the SET/CMOS hybrid design also improves, but is still inferior to that of the SET-only design.

Based on these results, we conclude that at room temperature, SET-only designs should be used for local interconnect; for wire lengths greater than 1 mm, the SET/CMOS hybrid design is more energy efficient. As temperature increases, the thermal energy impact increases. As a result, the static power consumption of SETs increases. Therefore, the wire length at which the SET/CMOS design begins to outperform the SET-only design decreases as temperature increases.

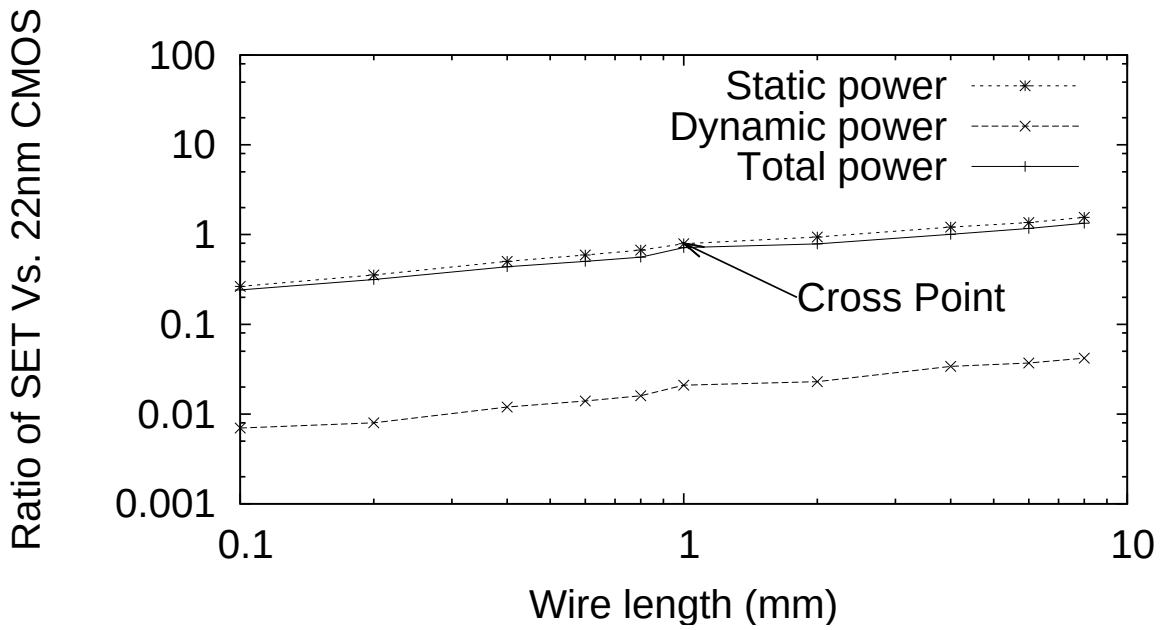


Figure 5.11. Power ratio of SET vs. hybrid interconnects for high performance for $C_\Sigma = e^2/(10k_BT)$.

Tables 5.6 and 5.7 illustrate two interesting trends for global interconnect. The power consumption of both the low-power and the high-performance $C_\Sigma \leq e^2/(40k_BT)$ hybrid SET/CMOS designs decrease with increasing temperature. At low temperatures, the output voltage ranges and driving currents for the SETs are small, increasing CMOS buffer static power consumption. The power consumption of the high-performance $C_\Sigma \leq e^2/(10k_BT)$ hybrid SET/CMOS interconnect design increases with increasing temperature because for this design, the SET output voltage range is small, resulting in increasing CMOS leakage at higher temperatures due to decreasing CMOS threshold voltage. The other designs were immune to this effect due to larger SET output voltage ranges.

5.4.1.3. Performance and Power Characterization of SET Non-Unate Logic. SETs support the efficient implementation of some non-unate arithmetic functions. We evaluate the power consumption and performance of an exclusive-or gate, a non-unate Boolean function widely

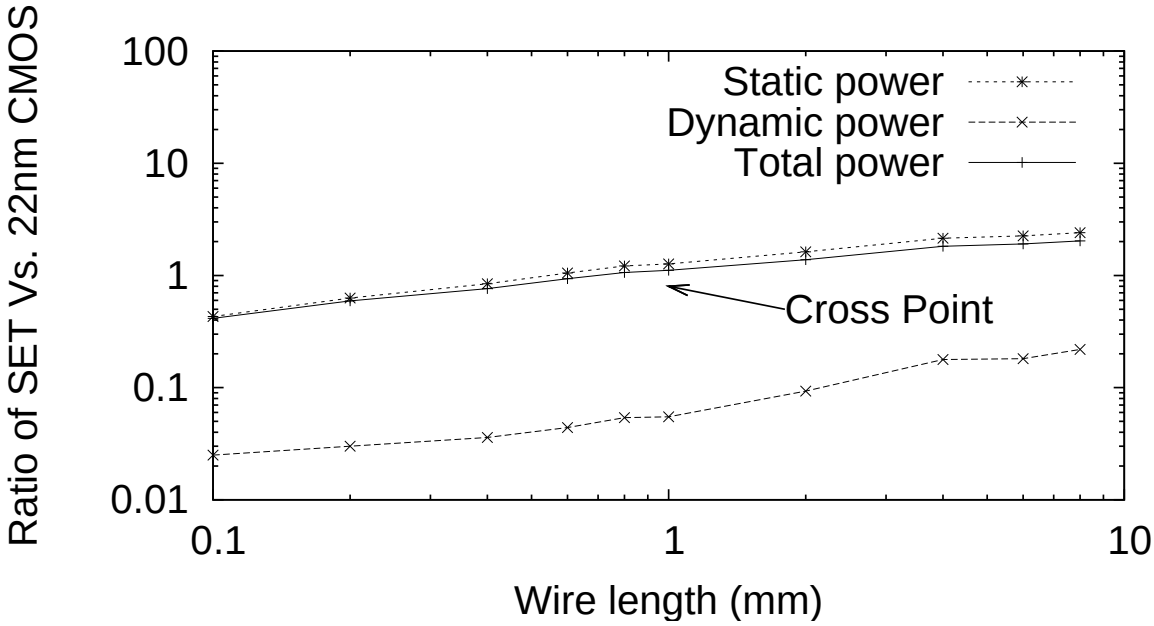


Figure 5.12. Power ratio of SET vs. hybrid interconnects for low power for $C_\Sigma = e^2/(10k_BT)$.

Table 5.8. Latency and Energy Improvement For Exclusive-Or Design

Performance setting	C_Σ constraint (F)	Performance improvement (%)	Energy improvement (%)
Battery	$e^2/(10k_BT)$	40.8	64.1
Battery	$e^2/(40k_BT)$	22.0	87.1
High	$e^2/(10k_BT)$	32.1	84.6
High	$e^2/(40k_BT)$	25.2	84.4

used in arithmetic logic, e.g., in addition and multiplication. We compared the two different implementations described in Section 5.3.2.3, the proposed SET-based design and the CMOS-style SET implementation. Figures 5.15, 5.16, 5.17, and 5.18 show the power and performance characterization of these two designs at the high-performance and the low-power settings at both $C_\Sigma = e^2/(10k_BT)$ and $C_\Sigma = e^2/(40k_BT)$ settings. These results clearly demonstrate the superior power consumption and performance of this design style, which is not possible using BJTs, CMOS, or threshold logic. Compared to the CMOS-style SET implementation, the design that

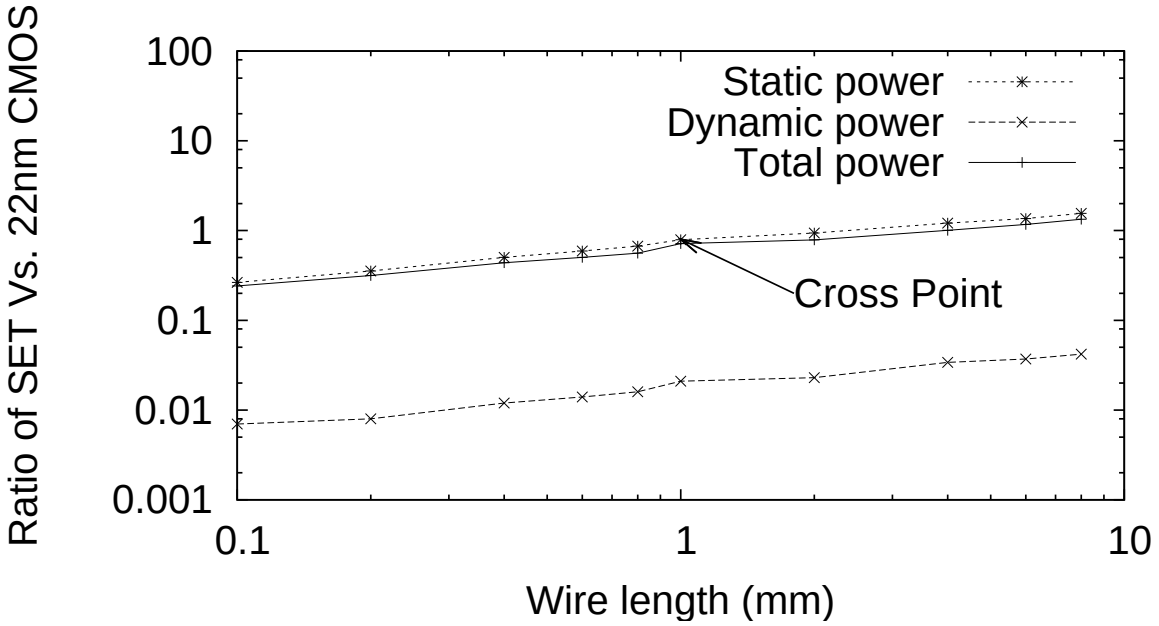


Figure 5.13. Power ratio of SET vs. hybrid interconnects for high performance for $C_\Sigma = e^2/(40k_BT)$.

exploits the periodic I–V curve of SETs achieves the latency and power consumption reductions indicated in Table 5.8, i.e., approximately a 25% reduction in latency and 75% reduction in energy consumption.

5.4.2. Characterization of High-Performance and Battery-Powered Embedded Applications

In this section, we characterize the performance and power consumption of IceFlex when used to implement numerous general-purpose and application-specific processor cores. We evaluate the suitability of IceFlex for use in both portable battery-powered and high-performance embedded systems by determining is performance and energy efficiency when used to implement the processor cores described in Table 5.9. We divided processors into two categories

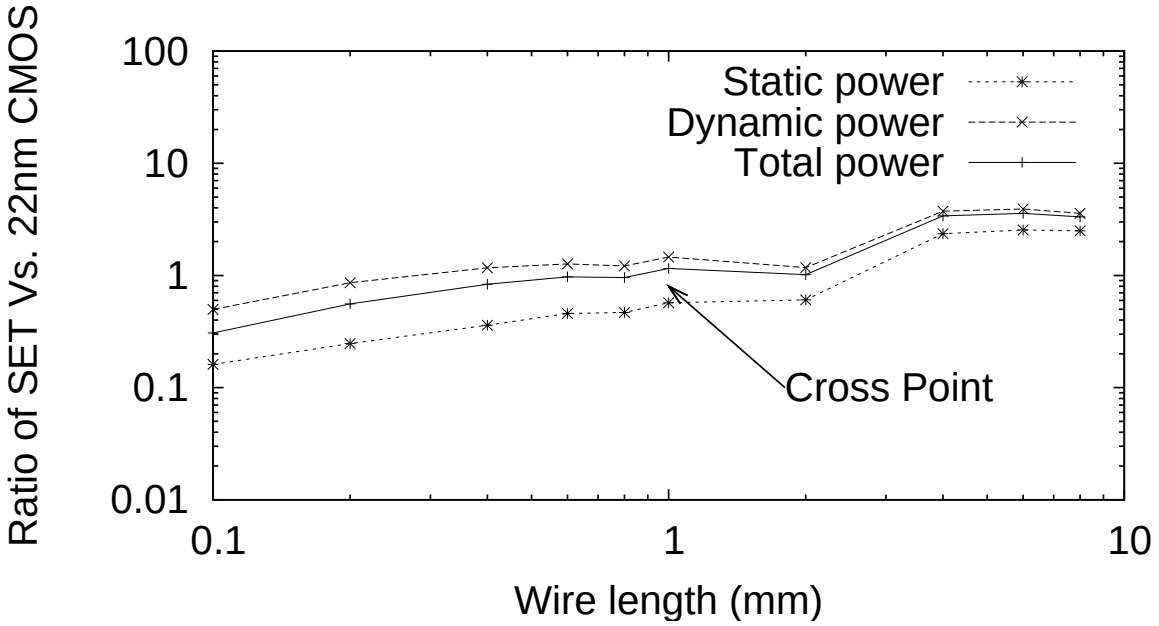


Figure 5.14. Power ratio of SET vs. hybrid interconnects for low power for $C_\Sigma = e^2/(40k_BT)$.

for the convenience of the reader. In fact, one might potentially use a "High performance" processor in a battery powered application, or vice versa.

The Xilinx Virtex-II XC2V2000 FPGA is used as a base case for comparison. Each application is synthesized with Xilinx ISE to determine the number of required LUTs, maximum frequency, and power consumption, using a switching probability of 10% [**160**] and a 65 nm feature size. Then, we scale the FPGA synthesis results into a 22 nm process based on HSPICE predictive technology model simulation results for the two technologies [**161**]. We used FPGA synthesis software to estimate the number of IceFlex SELBs required. 16-entry Virtex-II LUTs were used due to their functional (but not structural) similarity to IceFlex SELBs. For each design, the maximum frequency for IceFlex was determined by multiplying the number of combinational SELBs along the longest combinational path by the delay of an IceFlex SELB plus the delay of a local interconnect. The Xilinx ISE synthesis software did not use global interconnects

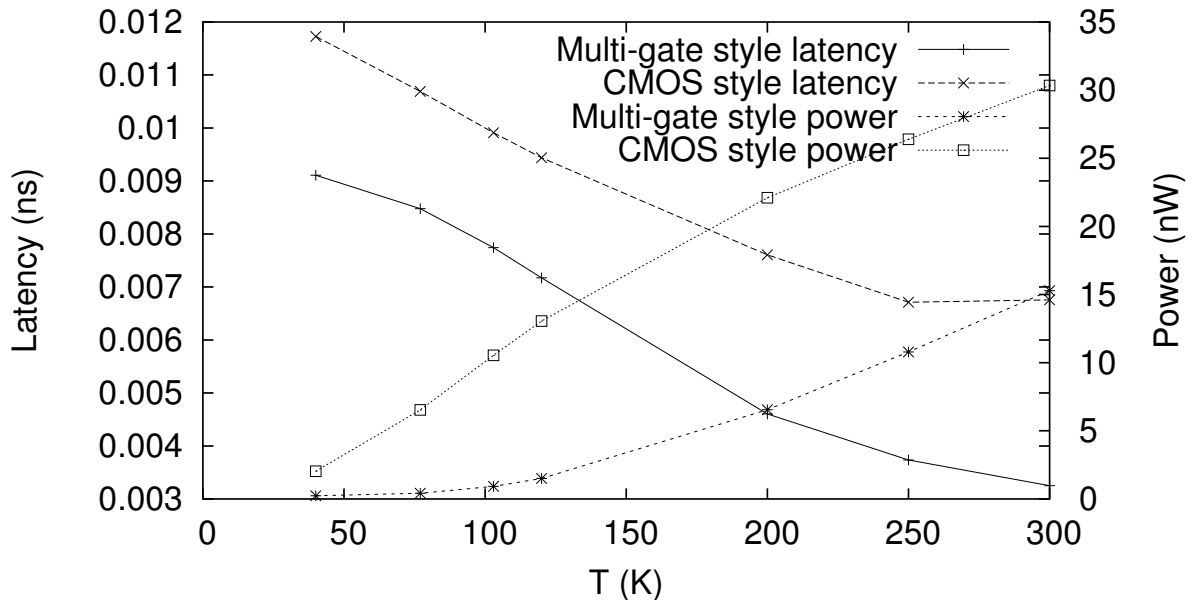


Figure 5.15. Performance and power characterization of exclusive-or logic for high performance for $C_\Sigma = e^2/(10k_BT)$.

for any of the synthesized processors. IceFlex power consumption was computed by taking the sum of the power consumptions of all components at the maximum operating frequency.

Figures 5.19, 5.20, 5.21, 5.22, and Tables 5.10 and 5.11 show the operating frequencies and energy efficiency in Joules per clock cycle of the XC2V2000 and various versions of IceFlex for each benchmark application. As described in Section 5.3.1.5, recent progress in fabrication is reducing the severity of the random background charge problem. If that work is successful, it may be less critical to use redundancy and majority voting logic in IceFlex. Therefore, we show the characteristics of both the spatially-redundant and non-spatially-redundant versions of IceFlex.

5.4.2.1. Ultra-Low-Power Applications. The data in Tables 5.10 and 5.11 indicate that the non-redundant, room temperature, low-power version of IceFlex is suitable for use in applications such as sensor network nodes, if they can be fabricated with sufficiently small island

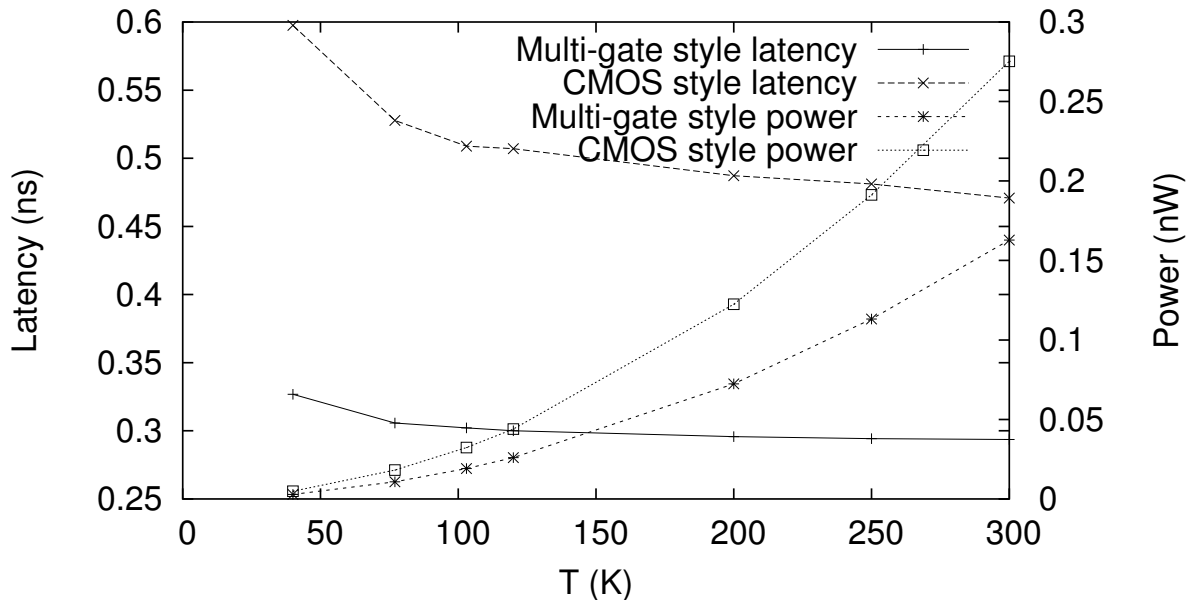


Figure 5.16. Performance and power characterization of exclusive-or logic for low power for $C_\Sigma = e^2/(10k_BT)$.

capacitances. In the following analysis, we shall focus on the AVR core, which is representative of the most commonly-used processor for sensor network nodes. Our results indicate that, even when adhering to the conservative $C_\Sigma \leq e^2/(40k_BT)$ constraint, the power consumption of IceFlex is low enough to permit an AVR processor to operate at 4 MHz to the shelf life of a single AA battery, or to operate on energy scavenged from the environment.

Alkaline AA batteries typically have 2,800 mAH of energy and nominal operating voltages of 1.5 V, i.e., they can deliver approximately 15,000 J. A low-power IceFlex AVR implementation running at 4 MHz consumes approximately 200 µW, permitting it to run for 20 years on one AA battery, i.e., longer than the shelf life of most such batteries.

If we assume an energy scavenging volume of 5 cm^3 and use Roundy's power densities of 4 µW/cm^3 for indoor solar energy, 200 µW/cm^3 for vibrations, 10 µW/cm^3 for daily temperature variation, and 0.003 µW/cm^3 for acoustic noise at 75 dB [**162**], we find that one sensor network

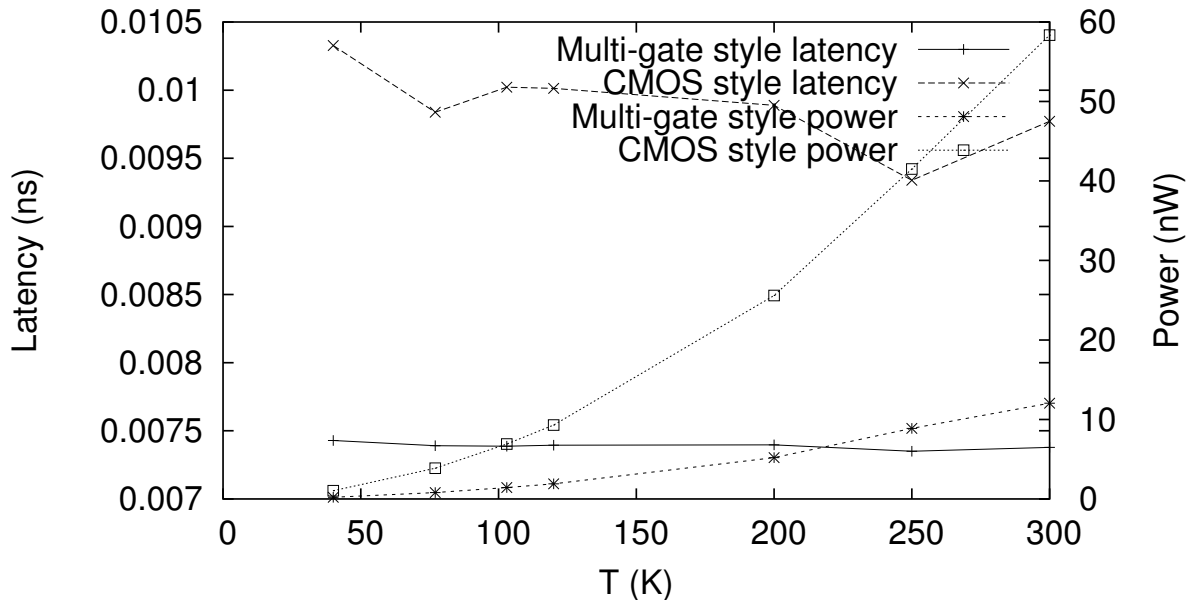


Figure 5.17. Performance and power characterization of exclusive-or logic for high performance for $C_\Sigma = e^2/(40k_BT)$.

node is capable of scavenging enough energy to power an IceFlex AVR processor running at the maximum clock frequency from vibrations or daily temperature variation, at 3.7 MHz from indoor solar energy, and at 2.8 kHz from 75 dB acoustic noise. However, SET circuits that operate at room temperature and adhere to the $C_\Sigma \leq e^2/(40k_BT)$ constraint will rely on features with sizes approaching (but not crossing) physical limits. Although the use of SETs in battery-powered applications has potential, it depends on the solution of formidable fabrication challenges or the development of compact, low-power cooling methods.

5.4.2.2. Energy-Efficient High-Performance Applications. We can draw the following general conclusions from Figures 5.19, 5.20, 5.21, and 5.22, as well as Tables 5.10 and 5.11. For a wide range of processor cores, the SET-based IceFlex architecture is capable of achieving energy efficiencies two orders of magnitude better than 22 nm CMOS-based FPGAs. Peak frequencies ranging from 200 MHz to 2 GHz are maintained for all processors.

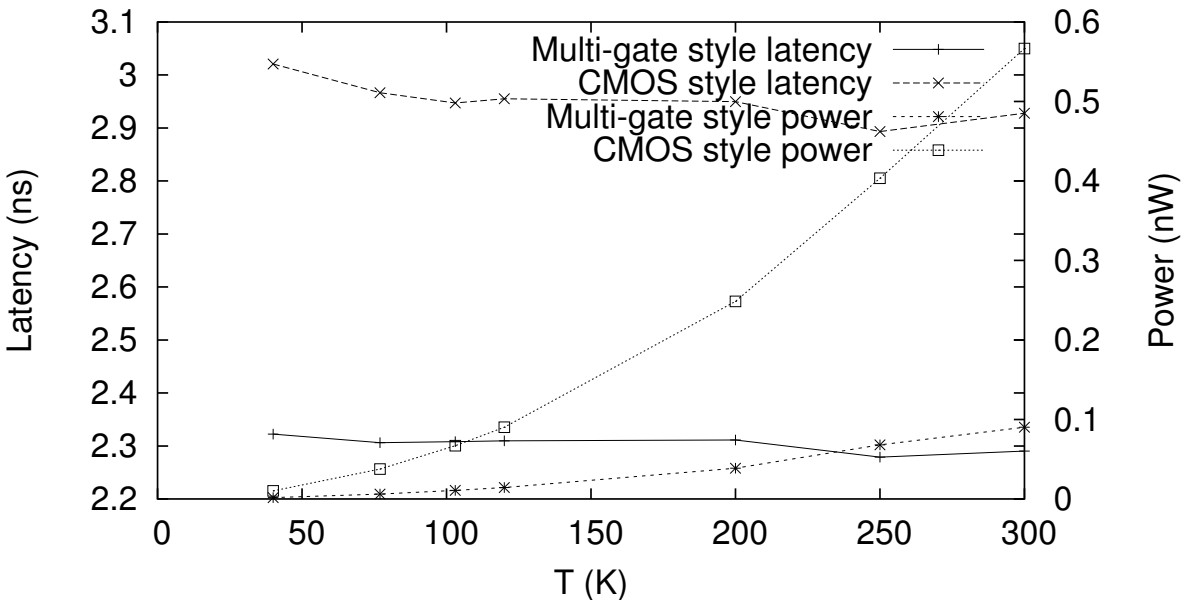


Figure 5.18. Performance and power characterization of exclusive-or logic for low power for $C_\Sigma = e^2/(40k_BT)$.

One might expect the high-performance version of IceFlex to consistently achieve higher frequency but require more Joules per clock cycle than the low-power version of IceFlex. However, it typically requires slightly fewer Joules per clock cycle, as well. Joules per clock cycle are computed at the maximum clock frequency of each processor. This has the effect of reducing the impact of static power consumption for processors with higher peak frequencies. If one must operate at a low frequency, the power consumption of the low-power version of IceFlex will generally be lower than that of the high-performance version. However, the reported values of Joules per clock cycle at maximum frequency have interesting implications. Although SETs have extremely low power consumption, at room-temperature a large percentage of this power consumption can be attributed to static power (see Figure 5.3). Therefore, for SET-based architectures that are operated at room temperature and have low performance requirements, it

Table 5.9. Characterization of Synthesized Processors

Type	Name	Description
Battery Powered	AES	AES (Rijndael) IP core
	AVR	ATMega103 microcontroller
	CORDIC	Coordinate rotation computer
	ECC	ECC core
	FPU	32-bit IEEE 754 floating-point unit
	RS	Reed Solomon encoder
	USB	USB 2.0 function
	VC	Video compression systems
High Performance	ARM7	Power-efficient RISC CPU
	ASPIDA DLX	Synchronous / DLX core
	Jam RISC	A five-stage pipeline RISC CPU
	LEON2 SPARC	Entire SPARC V8 processor
	Microblaze	RISC CPU
	miniMIPS	MIPS I clone
	MIPS	MIPS processor
	Plasma	Supports most MIP I opcodes
	UCore	MIPS I integer only clone
	YACC	MIPS I clone

will generally be more energy efficient to operate the device at high frequency and periodically enter a power-gated sleep mode than to continuously operate at a low frequency.

In high-performance applications for which parallel computation is appropriate, improved energy efficiency can be traded for improved performance with the same energy budget. For example, given a power budget of 125 mW and $C_\Sigma \leq e^2/(40k_BT)$, one could use one LEON2 SPARC implemented with an FPGA and running at 85 MHz or 5 LEON2 SPARCs implemented with the high-performance variant of IceFlex and operating at 1,025 MHz. This implies an overall performance 60× higher than that of the FPGA version. Taken to its logical extreme, assuming a power budget of 100 W and one instruction per cycle, one could execute 4.8 Terra IPS. These numbers are intended to give the reader some indication of the potential to improve

Table 5.10. IceFlex Performance and Power Consumption at Room Temperature For $C_\Sigma = e^2/(10k_BT)$

Benchmarks	FPGA		IceFlex							
	22 nm CMOS Technology*		Non-Redundant Battery Powered		Non-Redundant High Performance		Redundant Battery Powered		Redundant High Performance	
	Freq (MHz)	E req. (J/cycle)	Freq (MHz)	E req. (J/cycle)	Freq (MHz)	E req. (J/cycle)	Freq (MHz)	E req. (J/cycle)	Freq (MHz)	E req. (J/cycle)
ARM7	26.3	2.96e-09	3.5	3.93e-11	394.2	3.25e-11	3.3	2.85e-10	379.3	88.31
ASPIDA DLX	125.7	8.86e-10	20.4	4.49e-12	2325.6	3.72e-12	19.4	3.23e-11	2237.6	58.92
Jam RISC	95.8	8.92e-10	22.7	2.61e-12	2584.0	2.16e-12	21.5	1.90e-11	2486.3	38.50
LEON2 SPARC	85.9	1.88e-09	15.7	1.73e-11	1788.9	1.44e-11	14.9	1.27e-10	1721.3	177.77
Microblaze RISC	115.1	7.28e-10	29.2	1.38e-12	3322.3	1.14e-12	27.7	9.76e-12	3196.6	25.44
miniMIPS	88.0	4.87e-10	17.0	6.93e-12	1938.0	5.74e-12	16.1	5.00e-11	1864.7	76.01
MIPS	80.4	1.02e-09	18.6	3.07e-12	2114.2	2.54e-12	17.6	2.22e-11	2034.2	36.76
Plasma	75.3	1.13e-09	15.7	5.07e-12	1788.9	4.20e-12	14.9	3.73e-11	1721.3	52.31
UCore	136.4	8.19e-10	22.7	3.88e-12	2584.0	3.21e-12	21.5	2.80e-11	2486.3	56.87
YACC	72.1	1.18e-09	34.1	2.19e-12	3876.0	1.81e-12	32.3	1.58e-11	3729.4	48.11
AES	205.3	3.43e-10	51.1	1.66e-12	5813.9	1.38e-12	48.4	1.20e-11	5594.1	54.98
AVR	71.9	2.67e-10	17.0	3.81e-12	1938.0	3.16e-12	16.1	2.76e-11	1864.7	42.04
CORDIC	271.8	1.37e-10	204.4	1.34e-13	23255.8	1.11e-13	193.7	9.23e-13	22376.4	16.85
ECC	39.1	4.91e-10	20.4	4.42e-12	2325.6	3.66e-12	19.4	3.00e-11	2237.6	54.78
FPU	28.4	1.00e-09	4.5	5.89e-11	516.8	4.87e-11	4.3	4.33e-10	497.2	175.56
RS	496.7	1.28e-11	102.2	3.13e-14	11627.9	2.59e-14	96.9	2.21e-13	11188.2	2.02
USB	171.6	3.24e-10	68.1	1.05e-12	7751.9	8.72e-13	64.6	7.47e-12	7458.8	45.46
VC	114.2	1.24e-09	40.9	7.42e-12	4651.2	6.14e-12	38.7	5.38e-11	4475.3	196.21
Avg. Energy Impr.				95.60×		115.65×		12.27×		15.27×

performance given a power budget. In practice some of this performance will be lost due to parallelization inefficiency and off-chip communication latency. A similar comparison can be used for the MIPS processor, for which IceFlex permits a 268× improvement in energy efficiency compared with an FPGA implementation.

5.4.2.3. Reliability. As described in Section 5.3.2.5, we have designed SET-only MVL circuitry to support spatial redundancy in IceFlex. The exact probability of each SET failing is currently unknown. Therefore, we characterized the reliability impact of a number of spatial redundancy configurations in Section 5.3.2.5. We have analyzed the performance and power of a configuration in which seven-way SELB spatial redundancy is used together with MVL

Table 5.11. IceFlex Performance and Power Consumption at Room Temperature For $C_\Sigma = e^2/(40k_BT)$

Benchmarks	FPGA		IceFlex							
	22 nm CMOS Technology*		Non-Redundant Battery Powered		Non-Redundant High Performance		Redundant Battery Powered		Redundant High Performance	
	Freq (MHz)	E req. (J/cycle)	Freq (MHz)	E req. (J/cycle)	Freq (MHz)	E req. (J/cycle)	Freq (MHz)	E req. (J/cycle)	Freq (MHz)	E req. (J/cycle)
ARM7	26.3	2.96e-09	2.0	5.47e-11	224.0	4.79e-11	1.8	3.93e-10	216.9	3.28e-10
ASPIDA DLX	125.7	8.86e-10	11.5	6.37e-12	1333.3	5.58e-12	10.8	4.46e-11	1279.8	3.72e-11
Jam RISC	95.9	8.92e-10	12.8	3.65e-12	1481.5	3.19e-12	12.0	2.61e-11	1422.0	2.18e-11
LEON2 SPARC	85.9	1.88e-09	8.8	2.39e-11	1025.6	2.09e-11	8.3	1.74e-10	984.4	1.45e-10
Microblaze RISC	115.1	7.28e-10	16.4	2.01e-12	1904.8	1.76e-12	15.4	1.35e-11	1828.2	1.13e-11
miniMIPS	88.0	4.87e-10	9.6	9.78e-12	1111.1	8.56e-12	9.0	6.90e-11	1066.5	5.76e-11
MIPS	80.4	1.02e-09	10.5	4.34e-12	1212.1	3.80e-12	9.8	3.06e-11	1163.4	2.55e-11
Plasma	75.4	1.13e-09	8.8	6.91e-12	1025.6	6.05e-12	8.3	5.12e-11	984.4	4.27e-11
UCore	136.4	8.19e-10	12.8	5.45e-12	1481.5	4.78e-12	12.9	3.87e-11	1422.0	3.23e-11
YACC	72.1	1.18e-09	19.2	3.08e-12	2222.2	2.69e-12	18.0	2.18e-11	2132.9	1.82e-11
AES	205.3	3.43e-10	28.7	2.34e-12	3333.3	2.05e-12	26.9	1.66e-11	3199.4	1.39e-11
AVR	71.9	2.67e-10	9.6	5.34e-12	1111.1	4.67e-12	9.0	3.81e-11	1066.5	3.18e-11
CORDIC	271.8	1.37e-10	114.9	2.05e-13	13333.3	1.79e-13	107.7	1.29e-12	12797.5	1.08e-12
ECC	39.1	4.91e-10	11.5	6.92e-12	1333.3	6.05e-12	10.8	4.22e-11	1279.8	3.52e-11
FPU	28.4	1.00e-09	2.6	8.02e-11	296.3	7.02e-11	2.4	5.94e-10	284.4	4.96e-10
RS	496.7	1.28e-11	57.5	4.61e-14	6666.7	4.05e-14	53.9	3.07e-13	6398.8	2.57e-13
USB	171.6	3.24e-10	38.3	1.53e-12	4444.4	1.34e-12	35.9	1.04e-11	4265.9	8.65e-12
VC	114.16	1.24e-09	23.0	1.04e-11	2666.8	9.10e-12	21.6	7.41e-11	2559.51	6.19e-11
Avg. Energy Impr.				68.58×		78.46×		8.64×		10.55×

that internally uses fine-grained four-way SET parallelism. The MVL fault-tolerance hardware delays are added to the delay of each SELB stage. Tables 5.10 and 5.11 shows the performance and power implications of this configuration. Although the impact on maximum frequency is low, the use of seven-way structural redundancy generally increases the power consumption by 5–10 times, depending on the processor core.

5.5. Conclusions

In this chapter, we have analyzed the impact of using SETs in architecture and circuit design; proposed IceFlex, a fault-tolerant, reconfigurable, hybrid SET/CMOS architecture for

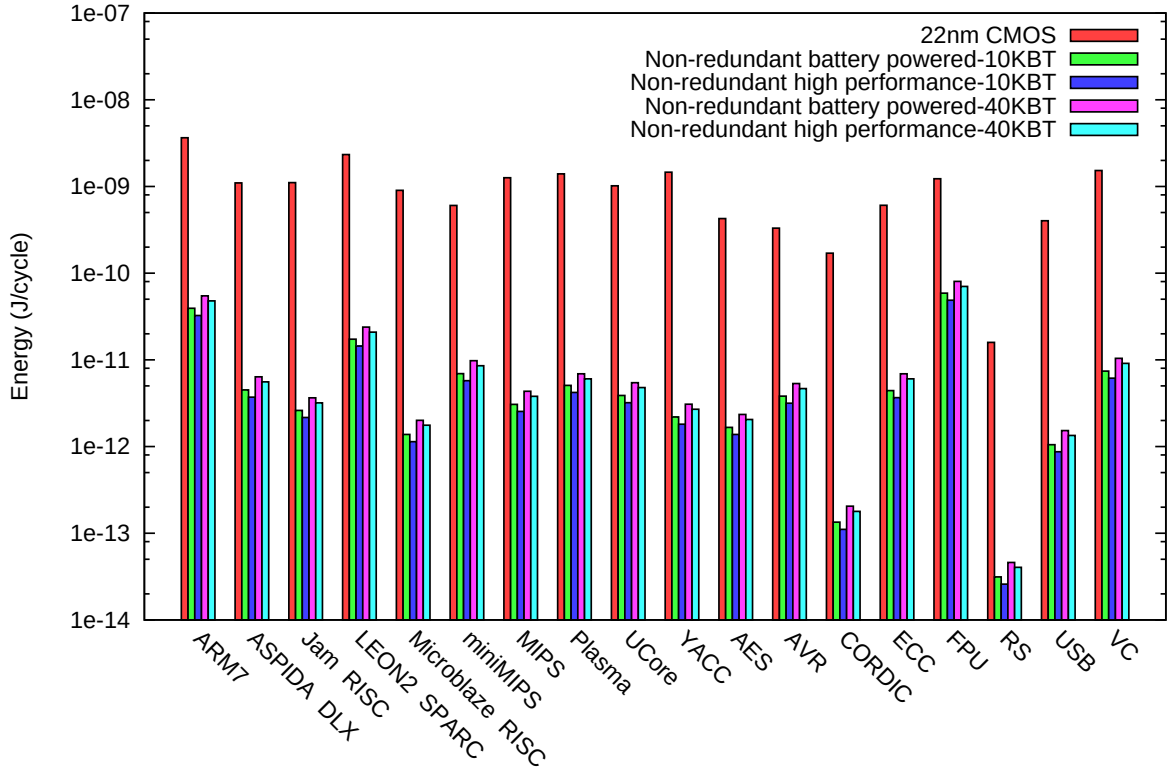


Figure 5.19. Energy Efficiency of Non-Redundant Design

use in high-performance and battery-powered embedded systems; and evaluated the energy efficiency, power consumption, and performance of IceFlex in these applications. Our results indicate that using SETs for computation poses many design challenges, but that many of these challenges can be solved with the proposed architecture and circuit design techniques. In addition, we find that SETs have some unique properties that permit significant improvements in circuit efficiency when compared with BJT, CMOS, and threshold logic based design. In summary, we find that a hybrid SETs/CMOS architecture has the potential to improve energy efficiency in battery-powered high-performance applications by two orders of magnitude compared with 22 nm CMOS while permitting operating frequencies that are as high, or higher.

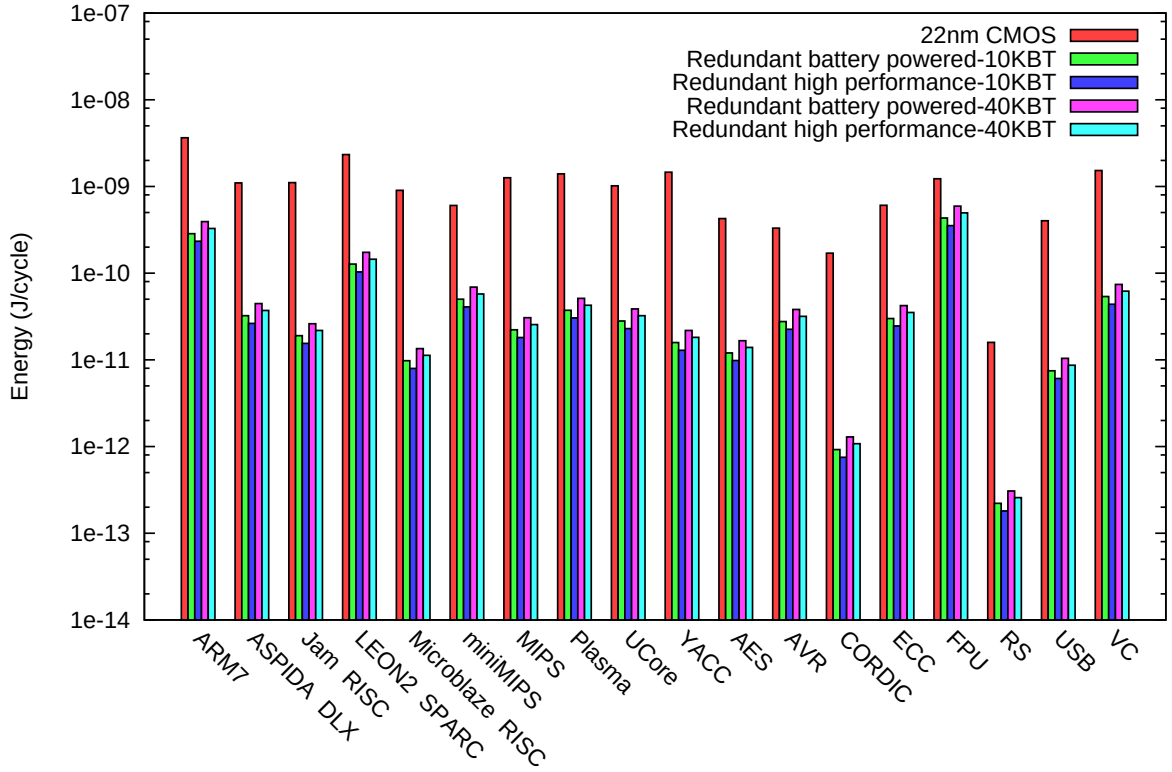


Figure 5.20. Energy Efficiency of Redundant Design

This improved energy efficiency can be traded for performance when operating within a power dissipation budget. Although they hold great promise, the practical use of SETs will require additional research into fault tolerance techniques, processing technologies, and novel circuit designs. In particular, the use of SET-based designs in portable applications will either require the fabrication of features with sizes approaching physical limits or the development of compact, energy-efficient technologies permitting operation below ambient temperature. It is our hope that this chapter provide a starting point for additional research in this area and reveals the potential advantages and challenges of SET-based architectures.

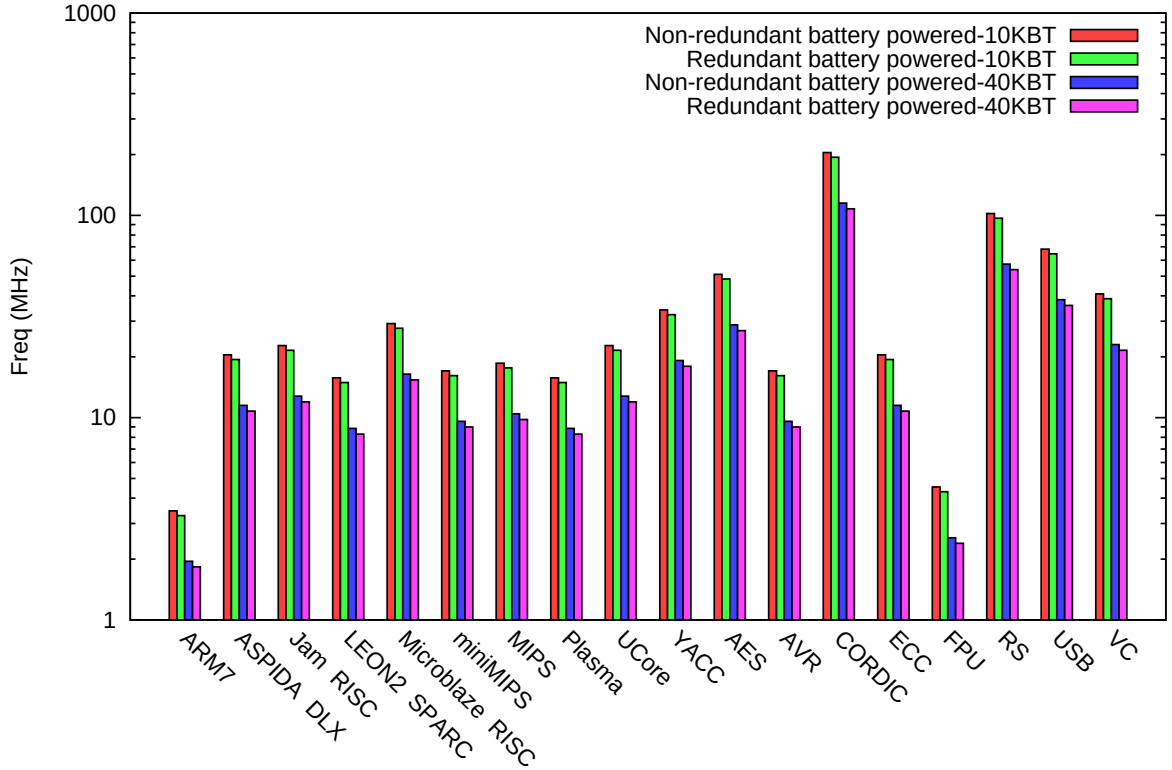


Figure 5.21. Frequency of Battery Powered Design

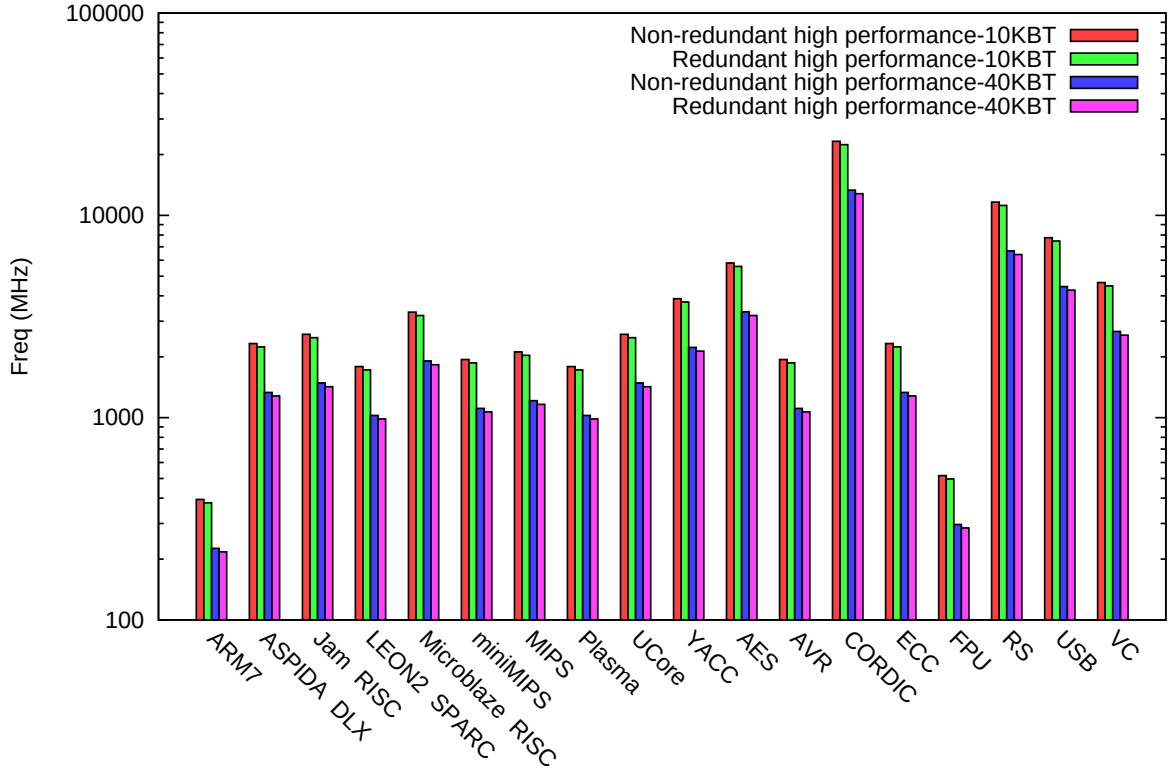


Figure 5.22. Frequency of High Performance Design

CHAPTER 6

Three-Dimensional Chip-Multiprocessor Run-Time Thermal Management

Three-dimensional (3D) integration has recently been proposed for chip-multiprocessor (CMP) design to improve communication latency and integration density. However, the stacked high power density layers of 3D CMPs increase the importance and difficulty of thermal management. In this chapter, we investigate the 3D CMP run-time thermal management problem and develop efficient management techniques. This work makes the following main contributions: (1) it identifies and describes the critical concepts required for optimal thermal management, namely the methods by which heterogeneity in both workload power characteristics and processor core thermal characteristics should be exploited; (2) it describes continuously-engaged hardware and operating system thermal management techniques that achieve better performance than state-of-the art techniques while honoring the same temperature bound; and (3) it presents a detailed evaluation of the proposed techniques using an integrated power, performance, and temperature full-system simulation environment. The software components of the proposed thermal management techniques have been implemented in the Linux 2.6.8 kernel. This source code will be publicly released. The analysis and techniques developed in this work provide a general solution for future both 3D and 2D CMPs. My major contribution to this chapter is on the design of 3D CMP architecture and technology, framework buildup of the full simulation system, and benchmark suites characteristics and generation.

6.1. Introduction

Control of power consumption and design complexity, as well as continued increases in integration density, are currently major concerns in computer architecture. As a result, microprocessor design is rapidly moving towards highly-scalable chip-multiprocessor (CMP) architectures. Today's mainstream microprocessors are multi-core [**13**, **14**, **15**, **16**, **17**, **18**]. The trend for future CMPs is to increase the number of on-chip cores: 80-core prototypes have recently been demonstrated by Intel [**163**].

Performance scalability is a major challenge in CMP design. Using the mainstream two-dimensional (2D) planar CMOS fabrication process, on-chip interconnect shows poor scalability in both performance and power consumption. Recently-developed three-dimensional (3D) integration is a promising solution to overcome the limitations of 2D technology [**23**, **164**, **165**, **166**]. By stacking multiple device layers connected through inter-die vias, 3D technology significantly reduces on-chip wire length (especially for global and semi-global wires), enables efficient interconnect and logic design, and further boosts logic integration density. This has motivated computer architects to adopt 3D technology to highly-scalable CMP architecture design [**164**, **167**, **168**, **169**].

Thermal issues are a large and growing concern for CMPs. Increasing chip power consumption and temperature affect circuit reliability (via electromigration, time-dependent dielectric breakdown, thermal cycling, etc.), power and energy consumption (via increased leakage power), and system cost (via increased cooling and packaging cost). The use of 3D integration magnifies power dissipation problems. Chip power density increases linearly with the number of vertically-stacked active circuit layers. In addition, the interconnect and bonding layers

used in 3D integration have low thermal conductivities, which further exacerbates thermal effects [**164**]. Temperature-related concerns that can be safely ignored in 2D planar process, such as self-heating, and temperature-induced performance degradation, become increasingly prominent in 3D CMPs. 3D integration holds promise but without solutions to the thermal problems it brings, 3D CMPs will be impractical.

Run-time thermal management techniques, such as dynamic voltage scaling, clock throttling, execution unit toggling, and workload migration, have been proposed in the recent past for 2D high-performance microprocessors [**170**, **76**, **171**, **172**, **173**, **1**]. Using these techniques, cooling solutions and packages needn't be designed for the worst-case power consumption. Cooling cost can thereby be significantly reduced. Most past work, however, cannot effectively optimize the performance and thermal tradeoff in 3D CMPs for the following reasons: (1) In current microprocessors, thermal management mechanisms are primarily used as a means to handle rare, worst-case processor power consumption and eliminate thermal emergencies. Although they can introduce performance overhead, they are rarely invoked. In contrast, the higher power densities of future 3D (and some 2D) CMPs will frequently require operation at or near thermal limits. Power should be viewed as limited resource and processor cores should spend carefully-budgeted amounts. Thermal management should be used as a constantly-engaged proactive run-time optimizer for CMP performance and thermal efficiency, instead of a reactive emergency measure. Therefore, its run-time performance and power overheads must be small; (2) 3D CMPs have heterogeneous power and thermal characteristics. On-chip processor cores have different cooling efficiencies. For instance, cores in the layers closer to the heat sink have higher cooling efficiencies than those farther from the heatsink. Inter-core thermal correlation is anisotropic. The thermal correlation between vertically-aligned processor cores

is much stronger than that between processor cores within the same layer. The power and thermal heterogeneity of 3D CMP poses unique challenges for run-time thermal management. Achieving optimal 3D CMP performance under a peak temperature constraint requires careful system-wide control of each processor core's performance and power consumption.

Our goal is to design 3D CMP thermal management techniques that deliver near-optimal performance and guarantee thermal safety. To reach this goal, we must first answer two main questions:

- **How does the move to 3D CMP architectures impact the run-time thermal management problem?** Our study identifies the challenges in 3D CMP thermal management. In addition, it determines the conditions necessary for optimal job assignment and power–thermal budgeting, and uses these to develop dynamic thermal management guidelines. We have found that the optimal run-time solution, i.e., maximal overall CMP performance subject to a temperature constraint, requires continuously-engaged, global power and thermal budgeting that considers the heterogeneous thermal characteristics of 3D CMPs.
- **Is it possible to design an efficient thermal management technique that yields near-optimal performance under peak temperature constraints for 3D CMPs?** We describe and evaluate a continuously-engaged, hardware–software run-time thermal management solution for 3D CMPs that delivers better throughput than existing work while guaranteeing thermal safety.

The M5 full-system simulation environment [**174**] is used to evaluate the proposed software-hardware based thermal management solution. Full-system multiprocessor simulation permits detailed evaluation of both the hardware and operating system (OS) components of the proposed

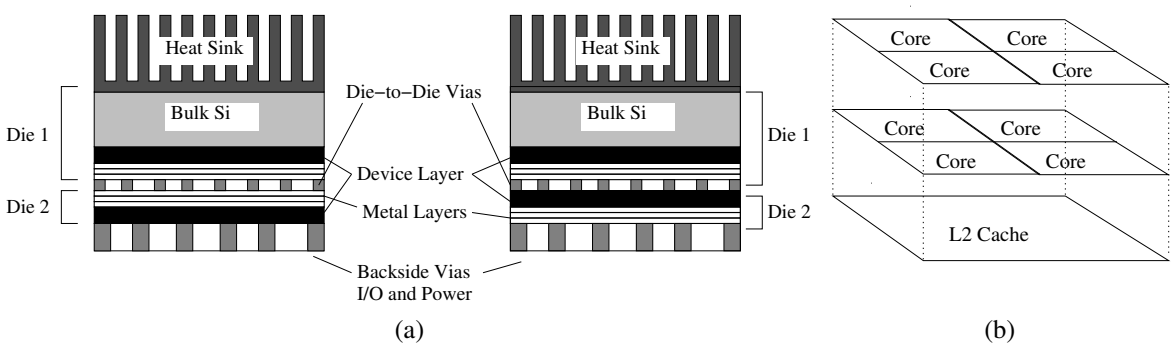


Figure 6.1. (a) Comparison of face-to-face (left) and face-to-back (right) configurations for two stacked dies and (b) 3D three stacked die floorplan used in this work.

thermal management techniques, making it possible to consider the effects of heterogeneity in workloads. Thermal management patches to the 2.6.8 Linux kernel will be publicly released.

6.2. Three-Dimensional CMP Technology

This section surveys the current status of 3D integration in microprocessor design and indicates the special thermal management challenges it will bring. A survey of related thermal management is given in Section 6.7.

6.2.1. Past Work on 3D Integration Technology

Several 3D fabrication technologies have been proposed and developed [**23**,**166**,**165**]. Topol et al. review the process steps and design aspects that were developed at IBM for 3D fabrication [**23**]. Tezzaron [**166**] and Samsung [**165**] develop 3D fabrication technologies and Intel is planning to use 3D integration in Terascale project [**163**].

As shown in Figure 6.1, there are two ways to stack two device layers: face-to-face and face-to-back. For designs with more than two layers, face-to-back bonding decreases worst-case inter-wafer via delay.

3D integration increases the importance of, and complicates, thermal management. The 2D heat flux density through the heatsink increases roughly linearly with the number of stacked wafers. As a result, unless per-layer power densities are greatly reduced, 3D CMPs will often operate near their thermal limits.

In addition to increasing the importance of thermal management, 3D design complicates thermal management policy design. In contrast with 2D CMPs, the temperatures of some pairs of 3D CMP processor cores, e.g., vertically-adjacent cores, are highly correlated. Moreover, in 2D CMPs, processor cores have similar thermal resistance to the ambient, and high thermal resistance to other cores. In 3D CMPs, core resistance to ambient and thermal interaction are highly-heterogeneous. For example, heat generated in cores farther from the heatsink must flow through more layers of silicon and polyimide bonding before reaching the heatsink.

6.2.2. Past work on 3D CMP Architecture

This section surveys recent research on 3D CMP design. Black et al. evaluate the performance improvement yielded by stacking memory and logic layers [**164**]. Healy et al. propose a microarchitecture-level floorplanning algorithm that works for both 2D and 3D ICs [**175**]. Kgil et al. propose an architecture in which processing core layers are vertically integrated with main memory consisting of multiple DRAM dies, permitting performance and power consumption improvements compared to 2D designs [**168**]. Li et al. propose a 3D topology that combines

the benefits of network-on-chip and 3D technology to reduce L2 cache latencies [**167**]. Tsai et al. explore cache implementation in 3D technologies [**176**].

Thermal issues are critical for 3D integration. Puttaswamy and Loh evaluated the thermal impact of 3D integration on high-performance microprocessors [**177**]. They also proposed a family of techniques that reduce 3D power density and assign more power to the die closet to the heat sink [**178**]. These approaches are principally applied at design time. Skadron et al. describe a compact thermal analysis technique that has been extended to support 3D integration [**76**]. Loi et al. studied processor and memory behavior under temperature constraints for 3D technology [**179**]. Link and Vijaykrishnan examine thermal effects in 3D technologies and show that vertical temperature differences are much smaller than horizontal temperature differences [**180**].

6.3. Thermal Properties of 3D CMPs

This section describes the properties of 3D CMPs. It also points out special thermal characteristics that must be considered to permit high-quality thermal management.

6.3.1. 3D CMP Power Model

In order to determine temperature profile, the power profile must first be known. We model both dynamic power consumption and leakage power consumption [**181**]. Dependence on voltage, switching activity, capacitance, and temperature are considered. These equations are used together with a Wattch-based EV6 power model [**182**] to determine the power consumption distribution among architectural units.

6.3.2. 3D CMP Thermal Model

There are two classes of thermal analysis problems: steady-state and dynamic. Dynamic thermal analysis considers transient effects, i.e., the resistance of materials to sudden temperature change, and is necessary to accurately model systems in which power consumption changes frequently. However, efficient and accurate dynamic thermal analysis is a hard problem. The 3D CMP and cooling structure may be decomposed into nearly isothermal (of uniform temperature) thermal elements and finite difference techniques are used to solve for temperature as a function of time and space. Fundamentally, the following equation must be efficiently solved for T.

$$\mathbf{C}dT(t)/dt + \mathbf{A}T(t) = Pu(t) \tag{6.1}$$

In this equation, given a system of N thermal elements, $\mathbf{C}$ is a an $N \times N$ matrix with thermal element heat capacities along the diagonal and zeros elsewhere, T is a length N thermal element temperature vector, t is time, $\mathbf{A}$ is an $N \times N$ thermal conductance matrix containing the conductances of adjacent elements at the corresponding row–column intersections and zeros elsewhere, P is a length N thermal element power vector, and $u(t)$ is a step function that changes from 0 to 1 at time t. Based on 3D power profiles and the 3D CMP thermal model, the temperature at each point in time may be computed. We use a frequency-domain moment matching technique for thermal analysis [**99**].

6.3.3. Special Thermal Properties of 3D CMPs

3D CMPs have a number of special thermal characteristics that must be considered during thermal management. When the ratio of thermal conductances to the ambient and the thermal

conductance between two processor cores is small, the temperatures of the two cores are highly correlated. In 2D CMPs the temperatures of all processor cores are moderately correlated, with only slight variation due to physical adjacency. As a consequence, local power management techniques can do a fair job at minimizing or bounding temperature. On the contrary, 3D CMPs exhibit two types of thermal heterogeneity that require special treatment during thermal management.

Let us now consider thermal heterogeneity in 3D CMPs more formally. Power consumed in one processor may influence the temperatures of other thermally-correlated processors. Equation 6.2 can be used as a starting point to compute the relative impact of a unit change in power consumption in one core on the temperature of another core.

$$\begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{n-1} \end{bmatrix} = \begin{bmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,n-1} \\ r_{1,0} & r_{1,1} & \cdots & r_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ r_{n-1,0} & r_{n-1,1} & \cdots & r_{n-1,n-1} \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{n-1} \end{bmatrix} \tag{6.2}$$

In this equation, $\mathbf{R} = \mathbf{A}^{-1}$. This matrix is conventionally called the thermal resistance matrix. However, it is more useful and accurate to view it as a power–temperature impact matrix. $r_{i,j}$ in $\mathbf{R}$ is the thermal contribution of a unit of power consumption on core j to core i's temperature. Assume for a moment that Core 0 is horizontally adjacent to Core 1 and vertically adjacent to Core 2. In this case, $r_{0,1}$ is a small value, on the order of 0.4 K/W; $r_{0,2}$ is 1.2 K/W approximately three times as large; and $r_{0,0}$ is also large, approximately 1.2 K/W or 1.4 K/W depending on whether the processor is adjacent to the heatsink or farther from it. These widely-varying power–temperature impact values make clear the heterogeneity of 3D CMPs. Vertically-adjacent processor cores have extremely high, e.g., 3–4$\times$ thermal correlation relative

to other processor pairs. Moreover, the thermal resistance to the ambient varies depending on the layer of a processor core. This heterogeneity requires global power-thermal budgeting.

6.4. 3D CMP Thermal Management

Below, we define and analyze the thermal management problem for 3D CMPs and determine the policies necessary for performance optimization under temperature constraints. This study will be used to guide the development of the run-time thermal management techniques described in Section 6.4.3.

6.4.1. Problem Definition

Given a 3D CMP with N on-chip processor cores, our goal is to maximize the CMP throughput under run-time thermal constraints. We consider the following two throughput metrics.

$$throughput_{CMP\_instruction} = \sum_{i=0}^{N-1} IPC_i \times freq_i, \tag{6.3}$$

the total number instructions executed by the CMP per second, i.e., the raw throughput and

$$throughput_{CMP\_frequency} = \sum_{i=0}^{N-1} freq_i, \tag{6.4}$$

the sum of the frequencies of the processor cores of the CMP, i.e., the multiprocessor analog of raw processor frequency. Run-time thermal safety requires that $T_i \leq T_{MAX}, \forall 0 \leq i \leq N-1$, i.e., the temperature of each processor core cannot exceed the maximum temperature constraint, T_{MAX}.

Conventional run-time thermal management techniques assume that temperature constraints will rarely be approached, and can be prevented by local emergency measures with little impact on performance. However, in future many-core 3D CMPs, processor cores will frequently operate at or near their temperature bounds. Therefore, 3D CMPs require continuously-engaged global power and thermal budgeting to maximize performance while maintaining thermal safety.

6.4.2. Conditions for Optimal 3D CMP Thermal Management and Derivations of Resulting Policy Guidelines

This section derives performance optimization guidelines for workload assignment and power–thermal budgeting.

Observation 1 *To maximize CMP throughput, processor cores should operate at different voltages and frequencies due to heterogeneous processor core thermal characteristics and heterogeneous run-time workloads.*

As described in Section 6.3, processor cores in a 3D CMP are thermally correlated. The temperature of each core i, is affected by the power consumptions of all cores, as follows:

$$T_i = r_{i,0} \times power_0 + \cdots + r_{i,N-1} \times power_{N-1} \leq T_{MAX} \tag{6.5}$$

where T_i is the temperature of processor core i, $r_{i,j}, \{i,j\} \in [0, N-1]$ is an inter-core thermal impact coefficient, which indicates the impact of a unit power consumption of core j on the temperature of core i. $power_j$ is core j's power consumption. N is the number of processor cores of the CMP.

To explain the relationship between performance and thermal impact, we define the following metrics: $TIP_{i,j} = dT_i/df_j$, the thermal impact on processor core i due to the increase of core

j's frequency, and $TIP_{i,j} = dT_i/df_j dIPC_j$, the thermal impact on processor core i due to the increase in core j's instructions per second.

Intuitively, *TIP* is the thermal cost per unit increase in processor core performance. Subject to a temperature bound, maximizing CMP performance requires that all the processor cores achieve the same thermal impact per performance improvement on the maximum-temperature core, i.e.,

$$TIP_{i,0} \equiv TIP_{i,1} \equiv \cdots \equiv TIP_{i,N-1} \tag{6.6}$$

Given that dynamic power consumption, $P_j = \xi_j V_j^2 f_j$, where V_j and f_j are the supply voltage and frequency of processor core j, yields

$$\begin{aligned} d(r_{i,0}\xi_0 V_0^2 f_0)/df_0 &\equiv d(r_{i,1}\xi_1 V_1^2 f_1)/df_1 \equiv \ldots \\ &\equiv d(r_{i,N-1}\xi_{N-1} V_{N-1}^2 f_{N-1})/df_{N-1} \end{aligned} \tag{6.7}$$

$$\begin{aligned} \frac{d(r_{i,0}\xi_0 V_0^2 f_0)}{df_0 dIPC_0} &\equiv \frac{d(r_{i,1}\xi_1 V_1^2 f_1)}{df_1 dIPC_1} \equiv \ldots \\ &\equiv \frac{d(r_{i,N-1}\xi_{N-1} V_{N-1}^2 f_{N-1})}{df_{N-1} dIPC_{N-1}} \end{aligned} \tag{6.8}$$

This result indicates that processor cores with heterogeneous power and thermal characteristics, i.e., different power–thermal impact coefficients $r_{i,j}$, running jobs with different switched capacitances ξ_j, should be clocked at different frequencies. Similar conclusions can be drawn when both dynamic and leakage power variants are considered.

Equation 6.6 provides the optimal-performance solution that satisfies the thermal constraint of processor core i. However, due to the heterogeneous power and thermal characteristics of

cores, this solution may not honor the temperature constraints of other cores. In addition, under different run-time workload distributions, Equation 6.6 produces different performances. Next, we will derive workload assignment as well as power and thermal budgeting policies for 3D CMPs. Workload assignment is used to optimize workload distribution for maximal performance optimization potential. Given a particular workload (i.e., thread distribution) power–thermal budgeting is used to control the voltages and frequencies of processor cores to optimize 3D CMP performance under temperature constraints.

As shown in Section 6.3, the inter-layer and intra-layer thermal characteristics of 3D CMP show distinct differences. This leads to different thermal management policies for inter-layer and intra-layer processor cores. In the following sections, we determine the conditions required for optimal 3D CMP thermal management and derive the resulting policy guidelines.

6.4.2.1. Inter-Layer Power–Thermal Budgeting. Inter-layer processor cores have heterogeneous thermal characteristics. We now derive thermal heterogeneity aware guidelines for power–thermal budgeting among vertically-aligned cores.

Guideline I *To maximize aggregate CMP frequency or instruction throughput, power–thermal budget assignment among inter-layer processor cores should follow Equation 6.6.* As shown in Section 6.3, among each group of vertically-aligned processor cores, the processor core i farthest from the heat sink has the lowest thermal efficiency and the highest temperature. Therefore, given the thermal constraint for processor core i, the performance-optimal voltage and frequency setup produced by Equation 6.6 also guarantees the thermal safety for other processor cores within the group. In other words, Equation 6.6 provides the performance-optimal power–thermal budget policy for inter-layer, vertically-aligned processor cores.

6.4.2.2. Inter-Layer Workload Assignment. Run-time workload has heterogeneous performance (IPC) and power (switched capacitance) characteristics. The following guidance provides the performance-optimal workload assignment policy among inter-layer, vertically aligned processor cores.

Guideline II *Given jobs with different switched capacitances, the maximal CMP throughput can only be achieved by maximizing the spatial heterogeneity of the switched capacitance during workload distribution. To maximize aggregate CMP frequency (Equation 6.4), jobs with higher switched capacitances should be assigned to cores with lower thermal efficiencies.*

Consider two vertically-aligned processor cores i and j. Core i is farther from the heat sink. Hence, $T_i > T_j$ and $r_{i,i} > r_{i,j}$. Assume the jobs running on both cores have the same switched capacitance ξ. To maximize aggregate CMP frequency, Equation 6.6 yields $f_i < f_j$ and $r_{i,i}power_i < r_{i,j}power_j$. Next, consider the effect of adjusting the workload assignment to increase the switched capacitance of the thread assigned to core i by $\delta\xi$ and decrease the switched capacitance of the jobs assigned to core j by $\delta\xi$. In other words, we assign jobs with higher switched capacitances to processor cores with lower thermal efficiencies. The peak temperature (core i's temperature) changes $\delta T_i = \delta\xi/\xi \times (r_{i,i}power_i - r_{i,j}power_j) < 0$, i.e., the temperature decreases. Therefore, both core frequencies can be further increased and aggregate CMP frequency can be further improved.

Guideline III *Given jobs with different IPCs, the maximal CMP throughput can only be achieved by maximizing the IPC heterogeneity during workload distribution. To maximize instruction throughput (Equation 6.3), jobs with higher IPCs should be assigned to cores with higher thermal efficiencies.* The derivation of this guideline is similar to that of the first two guidelines.

6.4.2.3. Intra-Layer Power–Thermal Budgeting. Intra-layer cores have mostly-homogeneous thermal characteristics with almost identical self power–thermal impacts (see Section 6.3), i.e., $r_{i,i} \approx r_{j,j}$, where core i and core j are in the same layer. In addition, the inter-core thermal impact is significantly lower than the self power–thermal impact of each core, i.e., $r_{i,i} \gg r_{i,j}$. We derive the following policies for intra-layer power–thermal budgeting and workload assignment.

Guideline IV *To maximize aggregate CMP frequency or instruction throughput, power–thermal budget and workload should be balanced among intra-layer processor cores.*

Consider two intra-layer processor cores i and j with $r_{i,i} \equiv r_{j,j} \gg r_{i,j} \equiv r_{j,i}$. Assume both cores are assigned the same voltage V, frequency f, and workload (ξ and IPC). Therefore, $T_i \equiv T_j$. Next, by adjusting the workload assignment, we increase the switched capacitance (IPC) of the jobs assigned to one core by $\delta\xi$ (δIPC) and decrease the switched capacitance (IPC) of the jobs assigned to another core. Since $r_{i,i}, r_{j,j} \gg r_{i,j}, r_{j,i}$, the peak temperature of these two cores will increase. In other words, introducing intra-layer workload heterogeneity degrades CMP thermal efficiency.

The polices and guidances derived in this section will drive our design of a run-time thermal management solution to maximize thermally-safe 3D CMP performance.

6.4.3. ThermOS: 3D CMP Thermal Management

Based on the thermal management guidelines developed in Sections 6.4.2.1–6.4.2.3, we designed ThermOS, a hardware–software run-time thermal management solution that optimizes 3D CMP performance subject to a peak temperature constraint. The following sections explain its components.

6.4.3.1. Temperature Monitoring. ThermOS gathers CMP temperature profiles at run-time, which are used to guide thermal-aware workload migration as well as power–thermal budgeting. Either thermal sensors or online thermal analysis may be used for on-line temperature monitoring. Thermal sensors have been widely used in high-performance microprocessors [**183**, **13**]. Efficient and accurate software-based online thermal analysis techniques have also been developed [**76**].

6.4.3.2. Workload Monitoring. In addition to CMP temperature profile, ThermOS gathers run-time performance and power characteristics to guide job migration as well as power–thermal budgeting. A processor core's switched capacitance is a function of the capacitances of its functional units and the corresponding run-time switching activities resulting from its workload. Most modern processors provide hardware performance counters for monitoring specific events [**13**, **184**]. These performance counters can be used to inform accurate and efficient regression-based run-time performance and power models [**185**, **186**]. ThermOS uses this technique for linear regression estimation of run-time processor core switched capacitances. The model is developed offline and implemented in the OS. At run-time, each processor core's hardware performance counter values are gathered periodically, triggered by OS timer interrupts (every 1 ms in Linux 2.6.8 kernel) and used for switched capacitance estimation.

6.4.3.3. Distributed Thermal-Aware Workload Migration. ThermOS contains a distributed online workload migration technique to support performance optimization. The proposed technique follows the guidelines derived in Section 6.4.2 and carefully handles the inter-layer thermal heterogeneities of 3D CMP and workload switched capacitance heterogeneities. To optimize CMP instruction throughput, ThermOS migrates jobs with high IPCs to processor cores with higher thermal efficiencies. In contrast, to optimize aggregate CMP frequency, ThermOS

migrates jobs with high switched capacitances to processor cores with lower thermal efficiencies. A distributed technique that swaps neighboring jobs developed.

Consider two vertically-adjacent processor cores: i and j. Assume core i has higher thermal efficiency than core j. To optimize instruction throughput, ThermOS compares the jobs stored in each processor core's job queue. It first identifies the lowest-IPC job ($IPCMIN_i$) on core i and the highest-IPC job ($IPCMAX_j$) on core j. If $IPCMIN_i < IPCMAX_j$, ThermOS swaps the corresponding jobs. Intra-layer thermal heterogeneity and thermal correlation are small. Therefore, ThermOS balances the intra-layer IPC distribution to optimize instruction throughput or the switched capacitance distribution to optimize aggregate CMP frequency. Average IPCs or switched capacitance values of jobs on intra-layer adjacent cores are compared, swapping jobs to further balance the distribution. The proposed distributed thermal-aware workload migration technique has been integrated within the default Linux kernel workload balancing policy. In the current implementation, workload migration occurs every 20 ms.

6.4.3.4. Global Power–Thermal Budgeting. ThermOS dynamically adjusts the power–thermal budgets of processor cores to optimize 3D CMP performance. Following the guidelines in Section 6.4.2, ThermOS balances the power–thermal budget assignment among processor cores in the same layer. Equation 6.6 is used to guide inter-layer power–thermal budgeting. The leakage-temperature dependency introduces temperature variables on both sides of Equation 6.6. Solving this equation requires numerical iteration and detailed chip-package thermal analysis which are computationally intensive. To minimize run-time overhead, we have developed an hybrid offline/online budgeting technique.

Given the switched capacitance (or IPC) range of the workload, the optimal voltage and frequency settings for inter-layer, vertically-aligned processor cores are pre-computed. The offline component of the budgeting algorithm is iterative. During each iteration, given the IPC and the switched capacitance of each processor core, Equation 6.5 and Equation 6.6 are used to determine the optimal processor core power–thermal budgets. Thermal analysis is then used to estimate the 3D CMP thermal profile and update the leakage power estimate. This process iterates until the chip-package temperature profile converges, subject to feedback from temperature-dependent leakage power consumption. The final voltage and frequency configurations are stored in a look-up table for efficient use during online, run-time power–thermal budgeting. Given the number of processor layers L and the number of switched capacitance settings n, the lookup table has n^L entries. Increasing n, i.e., the resolution of the switched capacitance index, improves performance but increases storage overhead, as evaluated in Section 6.6.3.2. In ThermOS, run-time power–thermal budgeting is implemented in Linux kernel and invoked periodically. In the current implementation, periods ranging from 1 ms to 100 ms are supported.

6.4.3.5. Distributed Run-Time Thermal Management. ThermOS uses distributed run-time thermal management to honor the power and thermal budgets described in Section 6.4.3.4 and adhere to a temperature constraint. Periodically, each processor core adjusts its voltage and frequency based on its assigned power–thermal budget. However, transient variations may not be immediately detected by the OS. In order to honor the temperature constraint, ThermOS uses local dynamic voltage and frequency scaling (DVFS) and clock throttling to react to transient variation with lower latency than global power–thermal budgeting. DVFS has less performance

impact per unit power reduction than clock throttling; however, clock throttling generally allows lower-latency and finer-grained control. In ThermOS, local DVFS continuously tracks temperature changes. Clock throttling is used as a final defense to guarantee thermal safety.

6.5. Experimental Setup

This section describes the experimental setup used to evaluate the proposed 3D CMP dynamic thermal management techniques. We describe our simulation and OS infrastructure, 3D chip and package models, and benchmark suites.

6.5.1. Infrastructure

Performance and temperature estimation for 3D CMP architectures is challenging. Estimating spatial and temporal thermal profiles requires time-varying power profiles. This, in turn, requires timing and power analysis. To accurately estimate the run-time characteristics of 3D CMPs, we developed a full-system multiprocessor simulation environment with integrated processor performance, power, and thermal models.

6.5.1.1. Full-System Simulation Setup. We use the M5 Full System Simulator [**174**]. M5 provides a detailed, cycle-accurate out-of-order multiprocessor simulation mode and a faster functional simulation mode. We use a combination of full-system checkpoints and the functional simulation mode to boot the system and fast-forward past the initialization portion of our benchmarks. We then switch to detailed simulation mode to evaluate thermal and performance characteristics.

We added a Wattch-based EV6 power model to M5 [**182**], scaled to the 90 nm process. Our cache power model is based on CACTI [**187**]. Static power consumption was estimated using

Table 6.1. Design Parameters for Alpha 21264

Alpha 21264 Configuration (90 nm)	
Die size	$4.56 \times 4.56\,mm^2$
Frequency and Voltage	2 GHz, 1.2 V
Instruction Queue	64 entries
Functional Units	4IXU, 2FPU, 1BPU
Physical Registers	80 GPR, 72 FPR
Branch Predictor	1 K local, 4 K global
Memory Hierarchy	
L1 DCache/core	32 KB, 2-way, 64 B blocks, 3 cycle lat.
L1 ICache/core	64 KB, 2-way, 64 B blocks, 1 cycle lat.
Shared L2 Cache	16 MB, 8-way LRU, 64 B blocks, 25 cycle lat.

Table 6.2. 3D Package Setup

Layer	Thermal cond. (W/mK)	Heat cap. (J/m^3K)	Depth (μm)
Eff. Active Layer (Silicon)	160.11	1.66×10^6	50
Eff. Interface Layer (Polyimide)	6.83	3.99	10
Heatsink (Cu)	400	3.55×10^6	6,900
Thermal Grease [**188**]	3–5 (5 used)	4*	50

* From configuration used in HotSpot [**76**].

an area-based, temperature-sensitive leakage model [**93**]. A 3D frequency-domain dynamic thermal analysis package was used [**99**].

6.5.1.2. Processor Architecture. We evaluate a three-layer front-to-back CMP structure. As shown in 6.1(b), there are eight Alpha 21264 microprocessor cores in the top two layers, each layer contains four microprocessor cores. Layers are connected with polyimide glue. There is 50 μm of thermal grease between the heatsink and die. Parameters for thermal grease and interface material follow Samson et al. [**188**].

Each processor core has 64 KB each of L1 data and instruction cache. There is a 16 MB shared L2 cache on Layer 2 and 1,024 MB of main memory. A 90 nm technology is modeled. Details can be found in the Table 6.1 and Table 6.2.

The via density in a region follows: $\rho_{via} = nA_{via}/wh$ where n is the number of vias in the region, A_{via} is the cross section area of each via, w is the width of the region, and h is the height of the region. The relationship between via density and effective vertical thermal conductivity

follows:

$$K_{eff} = \rho_{via} K_{via} + (1 - \rho_{via}) K_{layer} \tag{6.9}$$

where K_{via} is the thermal conductivity of the via material and K_{layer} is thermal conductivity of the region without any vias. Here, the via is assumed to be cooper with a thermal conductivity of 400 W/mK. A typical via size is $15 \times 15\,\mu m^2$.

For the Alpha 21264, there are 587 package pins (389 die pins). Interconnect vias use 0.64% of the core area. This results in the effective bulk silicon layer and interface layer thermal conductivities reported in Table 6.2. There are three types of heat sink: extruded, folded-fin, and integrated vapor-chamber. In this chapter, we assume an extruded heat sink with a thermal conductivity of 400 W/mK [**189**].

6.5.1.3. Operating System. The ThermOS run-time thermal management algorithms are implemented within the Linux-2.6.8 kernel. We made two main changes to the kernel:

- *Performance-counter based power modeling:* We enable OS-level power estimation using performance counters. Hardware event counters of the sort typical for modern processors were added to M5. A regression-based power model was added to the OS [**185**].
- *Power–thermal budgeting, task migration, and thermal management:* The proposed power–thermal budgeting and thermal-aware task migration techniques are implemented in the Linux kernel. We modified M5 to support kernel control of DVFS and clock throttling as well as temperature monitoring through privileged machine registers.

Table 6.3. Benchmark Characteristics

Group	Name	Avg. IPC	Avg. Power (W)	Max. T	Max. δT
SPEC High IPC	gcc	3.36	14.67	60.34	2.56
	applu	3.13	14.37	59.14	2.28
	gzip	2.78	13.34	58.37	3.39
	mgrid	2.58	13.66	59.25	3.06
SPEC Low IPC	twolf	1.58	11.33	56.30	2.26
	parser	1.55	10.41	55.57	2.81
	vpr	1.47	10.63	55.93	2.88
	mcf	1.25	10.91	57.59	2.50
Media High IPC	gsmenc	3.10	13.50	58.46	0.29
	jpegdec	2.72	13.42	57.72	0.29
Media Low IPC	g721enc	1.94	11.91	57.26	0.25

Table 6.4. Benchmark Suites

Group	Filename	Clusters	Benchmarks
SPEC	hv-hipc	High T var., high IPC	gzip, mgrid
	lv-hipc	Low T var., high IPC	applu, gcc
	hv-lipc	High T var., low IPC	parser, vpr
	lv-lipc	Low T var., low IPC	twolf, mcf
	hv-mipc1	High T var., mixed IPC	gzip, parser
	hv-mipc2	High T var., mixed IPC	mgrid, vpr
	lv-mipc1	Low T var., mixed IPC	applu, mcf
	lv-mipc2	Low T var., mixed IPC	gcc, twolf
Media	media-hipc	High IPC	jpegdec, gsmenc
	media-mipc	Mixed IPC	gsmenc, g721enc

6.5.2. Benchmark Suites

Two benchmark suites were used: SPEC2000 and Media Bench. Phansalkar et al. did a detailed analysis on SPEC2000 and found that it can be divided into different groups based on several microarchitecture-independent metrics [**190**]. In order to build up a complete set of test cases for our proposed techniques, we selected two microarchitecture-independent metrics: IPC and expected temperature variation.

- *IPC:* IPC is relatively stable and is approximately linearly-related to power consumption. Power consumption, in turn, has a strong influence on temperature.
- *Expected temperature variation:* The main motivation of our work is to maximize performance subject to a temperature constraint. We have prepared a set of benchmarks with a wide range of spatial and temporal thermal characteristics.

Based on these metrics, the benchmarks were analyzed, yielding the results in Table 6.3. Dynamic power traces were gathered during 500 ms to determine average power consumption, the temporal average of peak temperature, and the maximum peak temperature variation. Benchmarks containing mixes of high and low thermal variation and IPC were used (see Table 6.4). Each test case contained eight copies each of two benchmarks.

6.6. Experimental Results

This section evaluates ThermOS, the proposed run-time thermal management solution for 3D CMPs. The following design metrics have been used.

- *Effectiveness:* The capability to maximize performance while guaranteeing thermal safety, which is accomplished by explicitly considering the power and thermal heterogeneity of 3D CMPs and
- *Efficiency:* In a thermal bound CMP, run-time thermal management needs to be constantly engaged to optimize system performance and thermal efficiency. Therefore, the run-time performance and thermal overhead introduced by its OS and hardware components must be small.

6.6.1. Comparison of ThermOS With Alternatives

In this section, we first contrast ThermOS with solutions used in existing processors. Then we provide a detailed quantitative comparison with the most recent state-of-the-art continuously-engaged thermal management technique.

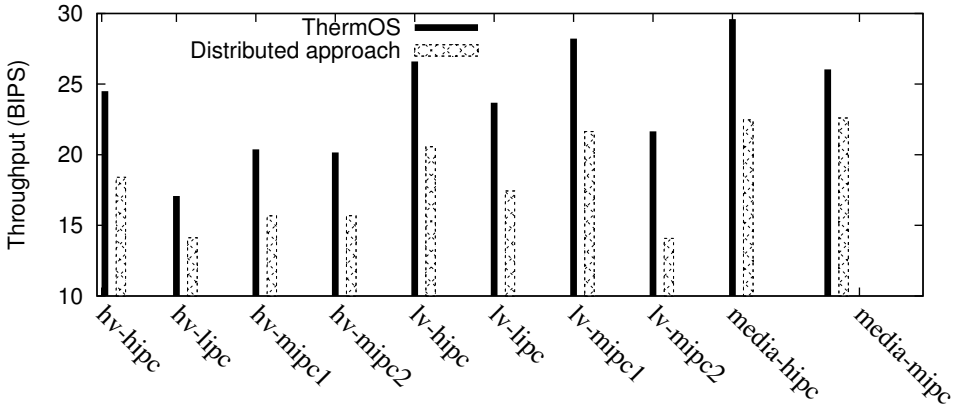


Figure 6.2. Comparison of ThermOS and Distributed Approach [**1**] CMP instruction throughput (defined in Equation 6.3).

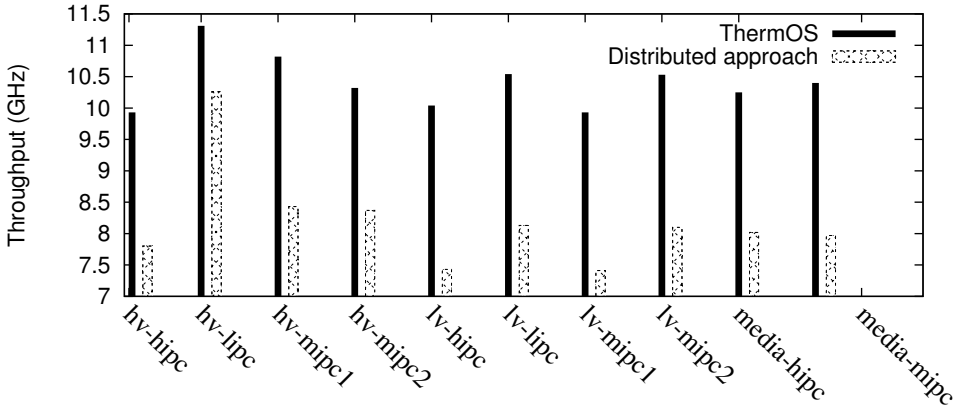


Figure 6.3. Comparison of ThermOS and Distributed Approach [**1**] aggregate CMP frequencies (defined in Equation 6.4).

Most thermal management techniques used in practice react to emergencies instead of being continuously-engaged. They detect dangerously-high temperatures and reduce power consumption, generally via hardware clock throttling. Such solutions are adequate when temperatures approach their limits only very rarely. However, high power densities and constraints on cooling costs require proactive thermal management. Some researchers have moved in this direction.

Donald and Martonosi [**1**] proposed a distributed continuously-engaged thermal management technique. Their approach is based on closed-loop control theory, and continuously adjusts the voltage and frequency of each processor core to maintain safe temperatures. Each core has its own controller and the controllers act independently. This permits significantly better performance than reactive approaches because DVFS can generally reduce power consumption by the same amount as clock throttling with less performance penalty. In fact, their results indicate that, compared with a stop-go based thermal control policy, distributed DVFS improves throughput by 2.5×. However, independent local control has limitations. The power consumed in one processor can impact the temperatures of other processors in nonuniform ways. As a result, continuously-engaged global control can permit better performance than continuously-engaged local control. This is especially true for 3D architectures, in which the power consumption of a particular processor core has great impact on the temperature of vertically-adjacent cores and relatively less impact on other cores.

ThermOS uses continuously-engaged, distributed global/local control to permit high performance, thermally-safe execution on 3D and 2D architectures. It has two primary differences with state-of-the-art temperature control techniques. First, it uses global power budgeting that takes into account the thermal interaction between processor cores. Second, it directs temperature-aware workload migration of threads among processor cores.

Figure 6.2 and Figure 6.3 show 3D CMP run-time throughput, in terms of instruction throughput (Equation 6.3) and aggregate CMP frequency (Equation 6.4), achieved by ThermOS and Donald's distributed approach. These results demonstrate that, compared to Donald's distributed approach, ThermOS improves instruction throughput by 30.91% on average (ranging from 15.22% to 53.79%) and improves aggregate CMP frequency by 27.63% on average

(ranging from 10.23% to 35.13%). The performance improvement can be explained as follows. In 3D CMPs, the strong thermal correlation among inter-layer vertically aligned processor cores has significant impact on the temperature of the processor layer farthest from the heat sink. Using the proposed power–thermal budgeting and thermal-aware workload migration techniques, ThermOS permits appropriate power budgets for each group of vertically-aligned processor cores. In addition, it uses DVFS to optimize the power–thermal efficiency of each processor core. Together, these techniques maximize overall throughput. The Donald's work, on the other hand, is a fully distributed, processor-local technique. Using this technique, each processor core regulates its power and performance to ensure local thermal safety without considering the thermal impact on neighboring cores. As a result, vertically-aligned processor cores are unable to efficiently share the power and thermal budget, which can reduce CMP performance.

In this section, we measure the performance permitted using a thermal management technique with near-optimal performance, but vulnerability to thermal violations due to transient changes in workload. We then show that the performance reduction resulting from the additional management techniques ThermOS uses to guarantee thermal safety is small.

ThermOS uses the thermal-aware workload migration and global power–thermal budgeting guidelines derived in Section 6.4.2. These techniques can potentially offer near-optimal run-time performance subject to a temperature constraint. However, they do not immediately react to transient workload variation occurring in individual processor cores, which may cause run-time thermal violations. ThermOS uses distributed run-time thermal control techniques to guarantee thermal safety, i.e., local DVFS and clock throttling dynamically adjusts the voltage and frequency of each processor core to eliminate thermal emergencies. Compared to DVFS, clock throttling is more responsive but has more performance impact. Therefore, in ThermOS,

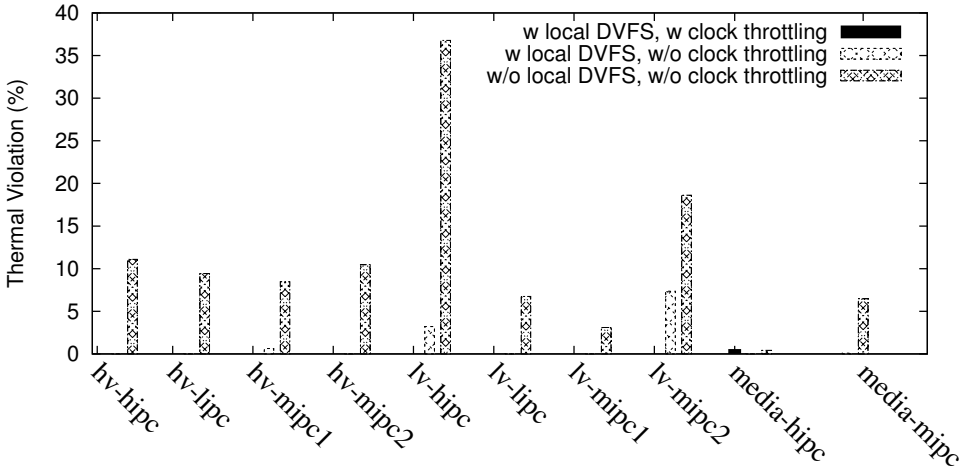


Figure 6.4. Reduction in thermal violations due to local DVFS and elimination of thermal violations due to clock throttling.

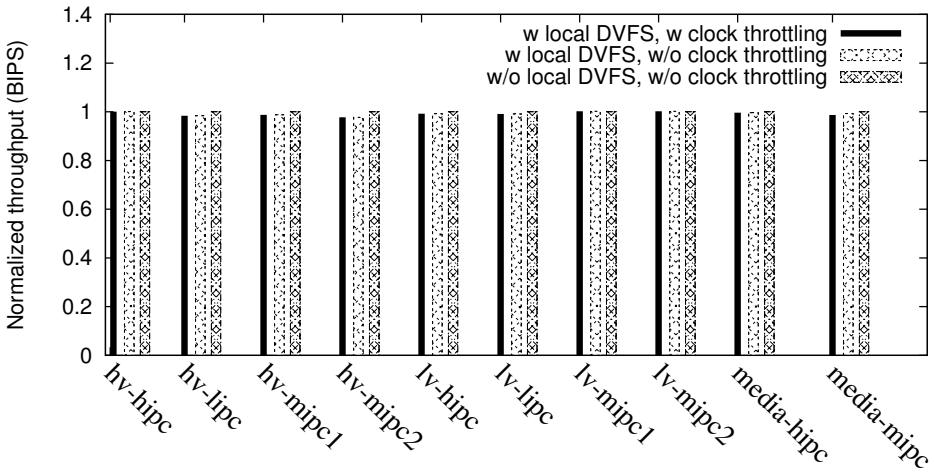


Figure 6.5. Negligible CMP instruction throughput reduction resulting from local DVFS and clock throttling.

DVFS is continuously engaged, and clock throttling is invoked only when local DVFS is incapable of guaranteeing thermal safety. These techniques, however, may deviate the run-time operations of processor cores from the performance-optimized guidance from power–thermal budgeting and thermal-aware workload migration, causing performance penalties.

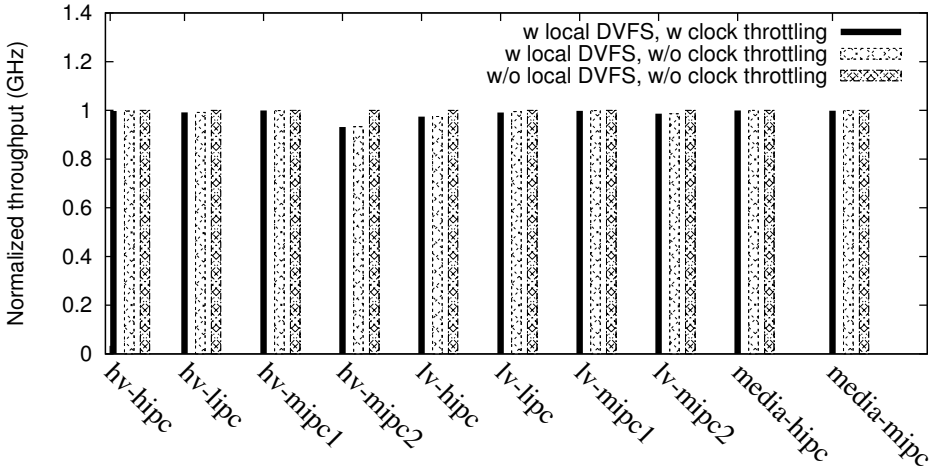


Figure 6.6. Negligible aggregate CMP frequency reduction resulting from local DVFS and clock throttling.

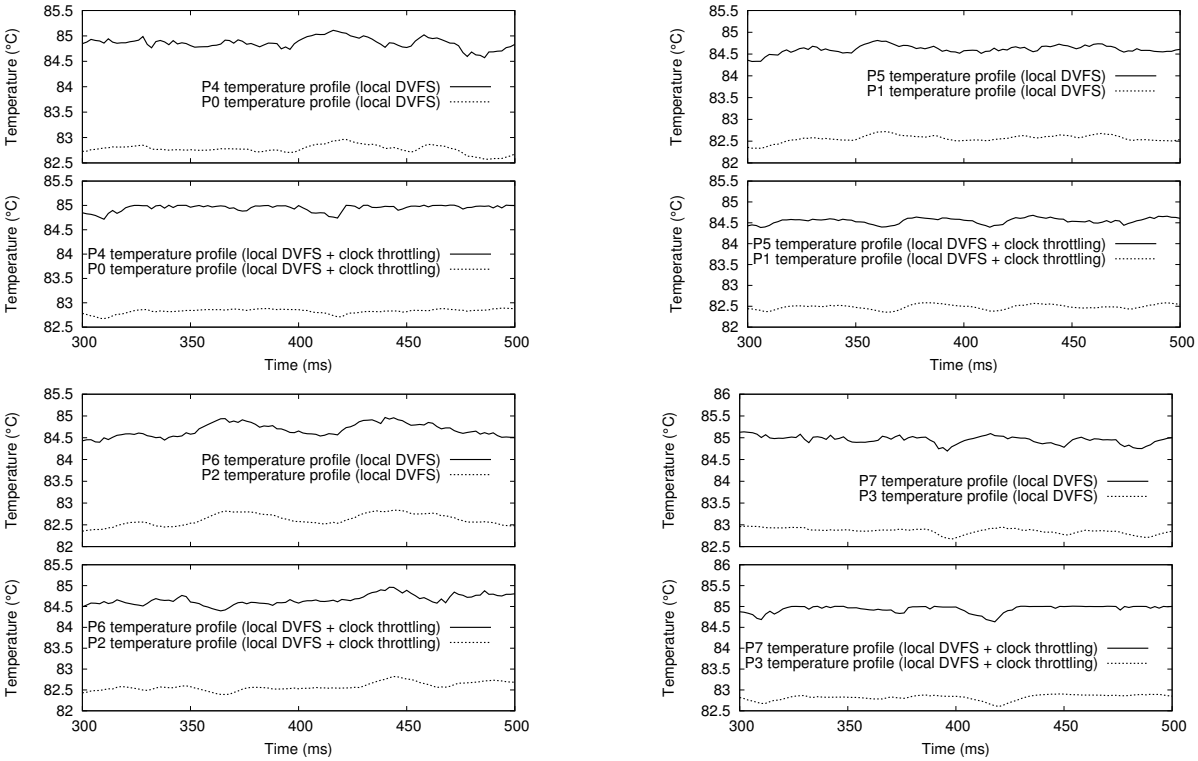


Figure 6.7. Temporal temperature variation for eight processor cores (P0–P7) running lv-mipc using local DVFS w.o. (top) and w. (bottom) clock throttling.

Figure 6.4 illustrates the levels of thermal safety achieved by various control techniques. As shown in this figure, when distributed control is disabled, the voltage and frequency of each processor core is solely controlled by global power–thermal budgeting, which does not consider the temporal workload variation within each processor core. This local workload variation can cause significant run-time power variation, and therefore thermal violations. Local DVFS can effectively track rapid workload variation occurring within each processor core and adjust voltage and frequency accordingly, thereby reducing the occurrences of run-time thermal emergencies. When clock throttling is also enabled, Figure 6.4 shows that processor thermal emergencies are completely eliminated.

To further illustrate the effectiveness of the distributed run-time control techniques, Figure 6.7 shows the run-time thermal profiles of eight on-chip processor cores when running the lv-mipc2 benchmark, with and without local clock throttling. Processors 0–3 are adjacent to the heatsink and processors 4–7 are closer to the printed circuit board. Local DVFS achieves a balanced CMP thermal profile, and run-time thermal violations (exceeding 85 °C, a predefined thermal threshold used in this experiment) occur only rarely. When both local DVFS and clock throttling are enabled, the temperature constraint is never violated.

Figures 6.5 and 6.6 indicate the performance penalty introduced by the distributed control techniques required to guarantee thermal safety. To help quantify the performance impact, we normalize the CMP throughput to the value achieved by global power–thermal budgeting and then evaluate the CMP throughput with local DVFS only and with both local DVFS and clock throttling. These results indicate that local DVFS degrades performance by 0.74% and 1.25% on average for instruction throughput and aggregate CMP frequency. In addition, since local DVFS is capable of eliminating most run-time thermal emergencies, clock throttling is rarely invoked.

As shown in these figures, enabling both local DVFS and clock throttling results in performance penalties of only 0.80% and 1.33% on average for instruction throughput and aggregate CMP frequency. In summary, the proposed distributed run-time thermal control technique achieves thermal safety with low performance impact.

6.6.2. Robustness to Changes in 3D Integration Process

In order to show the robustness of ThermOS to variation in 3D integration style, we evaluated the performance improvement when used for CMPs using front-to-back and front-to-front wafer integration (see Section 6.5.1). We simulated the proposed technique and the local distributed approach [**1**] for both integration styles using all benchmark mixes shown in Table 6.4. CMP instruction throughput and aggregate CMP frequency improvements were compared. The average CMP instruction throughput improvement was 30.91% for front-to-back integration and 26.40% for front-to-front integration. The average aggregate CMP frequency improvement achieved by ThermOS over the local distributed approach was 27.63% for front-to-back integration and 28.29% for front-to-back integration. For all combination of benchmarks and packages, the instruction throughput and aggregate CMP frequency improvements were greater than 7%. We can conclude that ThermOS permits substantial improvements in performance over the local, distributed technique for different 3D integration styles.

6.6.3. Scalability analysis of ThermOS

ThermOS uses distributed thermal-aware workload migration, global power–thermal budgeting, and distributed run-time thermal control techniques to optimize 3D CMP throughput and

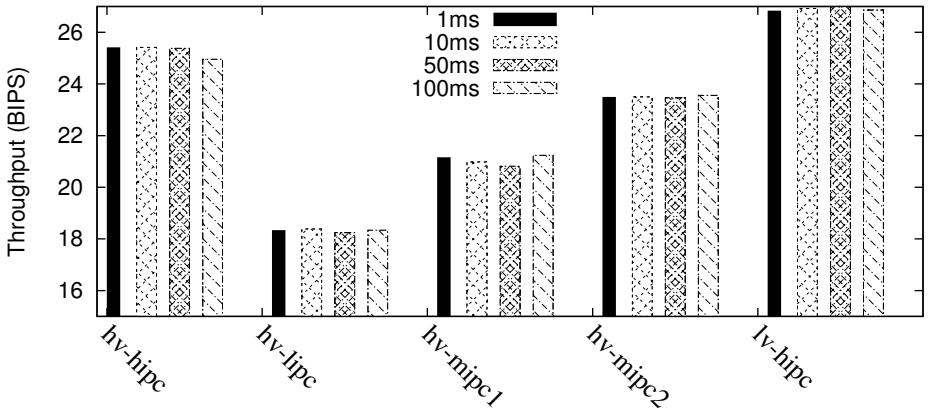


Figure 6.8. Global guidance interval impact on CMP instruction throughput.

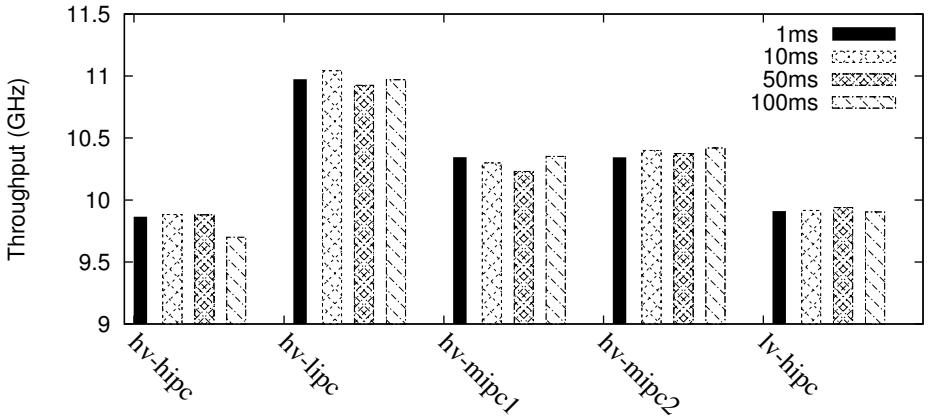


Figure 6.9. Global guidance interval impact on aggregate CMP frequency.

guarantee thermal safety. In contrast with purely local distributed techniques, run-time power–thermal budgeting is global. This might raise concerns about the scalability of ThermOS when used on many-core 3D CMPs. In this section, we evaluate the scalability of the proposed global power–thermal budgeting technique.

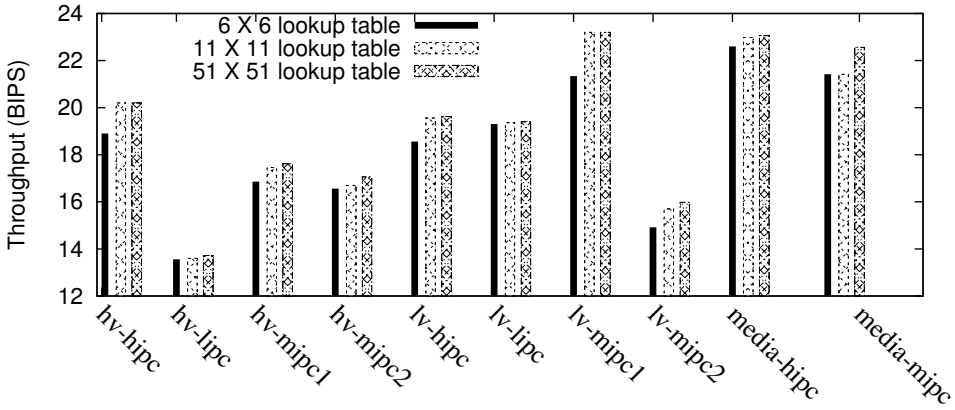


Figure 6.10. Lookup table size impact on CMP instruction throughput.

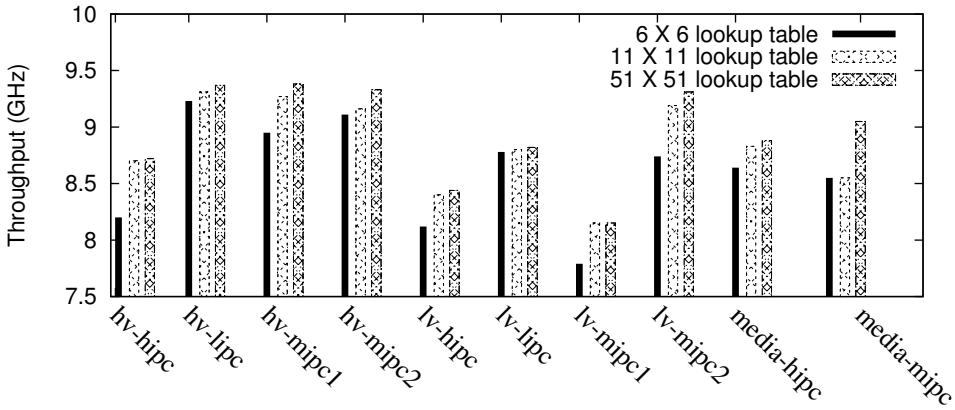


Figure 6.11. Lookup table size impact on aggregate CMP frequency.

6.6.3.1. Performance Impact. ThermOS periodically decides power–thermal budgets for processor cores. This involves inter-layer and intra-layer assignment. Run-time inter-layer assignment uses efficient table lookup. Intra-layer assignment uses an efficient homogeneous assignment policy, i.e., processor cores within the same layer are assigned the same power–thermal

budgets. In the current setup, i.e., an eight-core 3D CMP with 1 ms global guidance interval, detailed simulation shows that the overall run-time overhead introduced by global power–thermal budgeting is only 0.22%.

The run-time overhead of global power–thermal budgeting is linearly proportional to the run-time global guidance/budgeting interval. In general, shorter global guidance intervals can more accurately track run-time workload variation but may introduce more run-time overhead and communication contention when aggregating data from different CMP cores. It might therefore be useful to reduce this overhead by increasing the global guidance interval.

In the current setup, 1 ms guidance interval is used. This is frequent enough to allow adjustments in global power–thermal budget before temporal workload variation can produce large temperature changes, i.e., a higher frequency is unnecessary. To evaluate the impact of increasing global guidance interval on system performance, we choose run all four benchmarks with high workload variation from Table 6.4. One low-variation benchmark is also included for the sake of comparison. The results are shown in Figures 6.8 and 6.9. They indicate that, for guidance intervals up to and including 100 ms, ThermOS maintains nearly identical performance. Only hv-hipc experiences noticeable performance degradation. This benchmarks has the most significant temporal workload variation in the benchmark suite. However, changing the global guidance interval from 1 ms to 100 ms reduces CMP instruction throughput and aggregate CMP frequency by only 1.81% and 1.67%, respectively. We conclude that even if it were necessary to reduce global guidance interval by two orders of magnitude in order to maintain low global power–thermal budgeting run-time overhead in many-core 3D CMPs, there would be little reduction in thermally-safe performance.

6.6.3.2. Storage Impact. As described in Section 6.4.3.4, ThermOS uses an offline iterative budgeting algorithm to precompute some power–thermal budgeting decisions, which are stored as a lookup table in the main memory for efficient run-time usage. This lookup table has n^L entries. Each entry requires 4 B storage. L is the number of processor layers. It is expected that the number of processor layers in 3D CMP will be limited. n is the number of switched capacitance settings, which affects the power–thermal budgeting resolution. Higher resolution improves the accuracy of the run-time power–thermal budgeting decisions, but also increases the storage requirements for the table. In the current setup, we use a two-dimensional lookup table with 51×51 entries (10.4 KB) which provides sufficient resolution for accurate power–thermal budgeting.

It might be useful to decrease lookup table resolution for many-core systems in order to control storage overhead. We evaluated the impact of decreasing lookup table resolution on thermally-safe CMP performance by running all benchmark mixes using 51×51, 11×11, and 6×6 tables. As shown in Figure 6.10 and Figure 6.11, compared to the 51×51 lookup table, the 11×11 lookup table setting reduces the memory usage to 484 B, with average CMP instruction throughput and aggregate CMP frequency reductions of 1.16% and 1.19%, respectively. When the table is reduced to 6×6 entries, memory usage decreases to 144 B, with average CMP instruction throughput and aggregate CMP frequency reductions of 4.25% and 3.75%. We conclude that ThermOS requires little storage and that its performance degrades slowly with reduced lookup table size.

6.7. Related Work

Our work draws upon research in 3D integration technology, 3D microprocessor design, and microprocessor thermal management. This section surveys recent work in microprocessor thermal management. A survey of 3D technology and microprocessor design is given in Section 6.2. Initially, thermal control strategies were seen as a last resort that would be typically *engaged infrequently*. However, due to transistor densities and limitations in cool technology, thermal control will be *constantly engaged*. ThermOS is targeted for this future design paradigm.

In contemporary high performance designs, high power densities and limitations in conventional air cooling collectively create thermal hotspots on chip. These high temperature regions are problematic because many prominent device and interconnect failure mechanisms have an exponential dependence on temperature. Furthermore, high-performance chips in future technologies will have drastically shorter usable lifetimes if these thermal concerns are not addressed [**191**, **93**]. In addition, temperature also has negative effects on static power and maximum clock frequency due to subthreshold leakage and carrier mobility.

Brooks and Martonosi presented one of the first evaluations of DTM [**170**]. In essence, DTM allows designers to target their designs for a common case where the temperature is not severe. They instead allow run-time mechanisms to detect and resolve potential thermal emergencies. This yields better overall performance than pessimistically designing hardware which fits strictly within the thermal envelope.

Related work has examined use of local throttling policies to bound the peak temperature. Brooks and Martonosi used windowed power as a proxy for temperature and enforced thermal constraints through throttling techniques, most notably *fetch toggling*. In later work, Skadron et al. proposed control theoretic DTM [**192**] and developed thermal models to capture lateral

heat flow and nuances of the heat-spreader and packaging interface [**76**]. Yi et al. examined the impact of physical limitations (including power and temperature) on CMP architecture design [**193**].

For multi-core processors, migration strategies can produce additional benefits by distributing heat across the chip. Heo et al. propose activity migration that reduces the power density of a hotspot by relocating computation to another physical location on chip [**194**]. Powell et al. explore the benefit of OS management in the context of SMTs/CMPs [**171**]. They propose the *Heat and Run* strategy, in which the OS co-schedules and migrates SMT threads to maximize resource utilization before a thermal emergency arises and then migrates computation to an idle core. Kumar et al. examine hybrid hardware-software management that uses hardware performance counters to characterize thermal behavior and kernel support to schedule tasks [**186**]. They evaluate their mechanism on a real system with SMT support and find significant benefits from considering system-level effects which cannot be accounted for with pure hardware techniques. While we also take advantage of kernel scheduling and performance counters, Kumar's work does not address multi-core management – which we do. Recent work by Park et al. examines energy-performance tradeoffs in multi-threaded applications [**195**].

The recent work that is closest to our own is a paper by Donald and Martonosi which examines CMP thermal management with distributed control-theoretic core management and a global controller which guides migration [**1**].

Our work differs in several respects. First, our work integrates continuously-engaged thermal management strategies as a part of normal operation. This allows ThermOS to be applicable for near future technologies where temperature regulation is an integral part of normal operation. Second, we evaluate DTM in the context of 3D CMP architectures, a relatively new

avenue for microarchitects. In our problem domain, the global guidance and notion of a power budget are particularly beneficial due to the heterogeneous thermal characteristics of processor cores and workload power consumption. Consequently, distributed approaches such as those proposed by Donald may not perform as well in a 3D architecture. By matching core cooling characteristics, application features, and voltage levels we can improve performance by limiting throttling and migration. We believe that we are the first to examine workload scheduling techniques attuned to the cooling heterogeneity in 3D architectures. Finally, we evaluate our proposed policies in a full system simulator, hence our experimental setup allows us to account for the overhead of DTM in the OS, including migration costs and context switches.

6.8. Conclusions

3D integration has the potential to significantly improve performance and integration density. However, it will also increase power density, thereby increasing the importance of using continuously-engaged thermal management techniques. It will also increase the heterogeneity in thermal interaction among processor cores. This requires careful consideration during thermal management policy design.

We have developed a mathematical formulation for optimizing workload assignment, power–thermal budgeting, and voltage mode selection for 3D CMP thermal management. This formulation has been used to develop a continuously-engaged hardware–software thermal management solution for 3D CMPs. The proposed solution has been implemented within the Linux kernel and has been evaluated using full-system 3D CMP and OS simulation. Our strategy outperforms a state-of-the-art continuously-active, but distributed, thermal management policy. In particular, given the same temperature bound, it permits a 30.91% average improvement in

instruction throughput and a 27.63% average improvement in aggregate CMP frequency. In summary, distributed local control is insufficient for CMPs in which thermal interaction among processor cores is heterogeneous, i.e., 3D CMPs. A combination of local and continuously-active global techniques is necessary.

CHAPTER 7

Conclusion and Future Work

Increasing IC power consumption raises average and peak temperatures. As projected by the International Technology Roadmap for Semiconductors (ITRS) [**2**], further process scaling will be bounded by power consumption and heat dissipation below 65 nm: it is critical to address the energy and thermal issues during on-chip system design to meet the urgent need of the semiconductor industry and enable future technology scaling. Therefore, in this dissertation, we presented several topics which are related to thermal-aware and power-aware design.

In thermal-aware design flow, designers must frequently trade off other design metrics, such as performance, area, and cooling costs, to meet tight thermal constraints. The interaction of power and thermal constraints with other design metrics further increases system complexity. Therefore, an efficient unified incremental high-level and physical-level synthesis algorithm was presented in Chapter 2, which enable the tight integration between high-level and physical-level design. Chapter 3 presented a thermal-aware high-level synthesis algorithm built upon the framework of Chapter 2. Temperature variations and hot spots account for reliability issue, most of which are due to electromigration, hot carrier effects, thermal stress, and oxide thermal breakdown. Power and thermal variation can also lead to significant timing uncertainty, requiring more conservative timing margins, thereby reducing performance. Therefore, reliability-aware synthesis flow was presented in Chapter 4. Recent developments in nanoscale devices open new alternatives for low-power embedded system design. Among these, single-electron

tunneling transistors hold the promise of achieving the lowest power consumption. Unfortunately, most analysis of SETs has focused on single devices instead of architectures, making it difficult to determine whether they are appropriate for low-power embedded systems. Therefore, a fault-tolerant hybrid SET/CMOS reconfigurable architecture was presented in Chapter 5. 3D CMOS technology has been developed to overcome the interconnect bottlenecks of 2D design, but at the cost of serious thermal problems due to significant increase of power density by stacking multiple active device layer together. Hence, in Chapter 6, 3D CMP framework was presented and thermal-aware management policy was evaluated built upon this framework.

There are several related issues of interests that can be explored in the future.

(1) Timing-driven high-level synthesis

A number of researchers have considered the impact of physical details, e.g., floorplanning information, on behavioral synthesis [**41**, **42**, **43**, **45**]. Interconnect and interconnect buffers are now first-order timing and power considerations in VLSI design [**51**]. This change has complicated both design and synthesis. It is no longer possible to accurately predict the power consumption and performance of a design without first knowing enough about its floorplan to predict the structure of its interconnect. For this reason, a number of researchers have worked on low-power interconnect-aware behavioral synthesis algorithms [**69**,**52**,**70**,**54**]. These approaches typically use a loosely coupled independent floorplanner for physical estimation. There are two drawbacks to this approach. First, the independent floorplanner may not be stable, i.e., a small change in the input netlist may result in a totally different floorplan. This results in a behavioral synthesis algorithm that bases its moves on cost functions without continuity. Second, even if the floorplanner is stable, creating a floorplan from scratch for

each behavioral synthesis move is not efficient, given the fact that the new floorplan has only small difference from the previous one. Recently, incremental floorplanning and synthesis [**55**] were used to tightly couple high-level and physical synthesis that dramatically improve their combined performance and quality [**8**]. Thermal problem also has been considered during high-level synthesis [**83**, **7**]. However, there are few works focusing on timing-driven high-level synthesis in the view of performance considering wire delay. Weng and Parker proposed a high-level synthesis system, which tried to reduce the total wire cost during scheduling and assignment procedure [**42**]. Cong et al. presented a regular distributed register microarchitecture for simultaneously scheduling with rebinding, and distributed control generation [**196**]. However, this is only based on proposed microarchitecture which is not general.

(2) Power-driven floorplanning/placement with static voltage scaling

Now that integrated circuit (IC) design has entered the era of nanometer-scale features, reducing and managing power dissipation are key challenges. Power consumption influences peak IC temperature and determines the battery lifespans of portable devices. Generally, power consumption comes from two sources, dynamic power and static power. While static power in current technology mainly comes from leakage current, dynamic power is consumed by transistors' switching activities. Researchers have proposed many circuit-level low power techniques. Static voltage scaling is a promising one, which provides fine-grain dynamic power and performance trade-off. However, this comes with the overhead of off-chip regulator and on-chip level shifter. Therefore, floorplanning is a good platform for implementing this technique while constraining the overhead introduced. Incremental technique can be used to tightly

combine voltage island assignment within floorplanning. In addition, compared with traditional path based slack distribution algorithm, network flow based method can be promising.

(3) 3D Thermal-aware floorplanning/placement

Three-dimensional (3D) chips have emerged as promising candidates to overcome the interconnect bottlenecks of nanometer scale design. However, it is expected that the benefits from this technology can potentially be offset by thermal considerations which impact chip performance and reliability. Therefore, one big issue of nanometer scale chip design is the thermal management. Temperature can be reduced through two directions given same power budget: 1) balance power consumption in time domain; 2) balance power consumption in physical space. Balancing the thermal profile from time domain can be formed as scheduling problem while the latter one can be implemented through floorplanning/placement.

On the other hand, temperature plays an important role in the dissipation of leakage current for transistors. From transistor physics, the subthreshold leakage current is sensitive to temperature. Leakage power has been shown to have a strong dependency on temperature especially for high temperature range [**197**]. Therefore, thermal-aware design is not only restricted to reduce the peak temperature, it also can consider reducing the total area of peak temperature, and reducing the total area with temperature above threshold. All these cases will help to reduce the leakage power, which will further reduce the peak temperature. This is different from previous work [**198**, **199**, **80**, **200**, **81**, **82**], which only focus on minimizing the chip peak temperature.

References

[1] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. Int. Symp. Computer Architecture*, June 2006.

[2] "International Technology Roadmap for Semiconductors," 2006. http://public.itrs.net.

[3] H. Wakabayashi, S. Yamagami, N. Ikezawa, A. Ogura, M. Narihiro, K. Arai, Y. Ochiai, K. Takeuchi, T. Yamamoto, and T. Mogami, "Sub-10-nm planar-Bulk-CMOS devices using lateral junction control," in *Proc. Int. Electron Devices Meeting*, pp. 989–991, Dec. 2003.

[4] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, pp. 23–29, July 1999.

[5] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of CMOS," in *Proc. Int. Electron Devices Meeting*, pp. 7–13, Dec. 2005.

[6] L.-T. Yeh and R. C. Chu, *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices*. New York, NY: ASME Press, 2002.

[7] Z. P. Gu, Y. Yang, J. Wang, R. P. Dick, and L. Shang, "TAPHS: Thermal-aware unified physical-level and high-level synthesis," in *Proc. Asia & South Pacific Design Automation Conf.*, pp. 879–885, Jan. 2006.

[8] Z. P. Gu, J. Wang, R. P. Dick, and H. Zhou, "Incremental Exploration of the Combined Physical and Behavioral Design Space," in *Proc. Design Automation Conf.*, pp. 208–213, June 2005.

[9] Y. Yang, Z. P. Gu, C. Zhu, L. Shang, and R. P. Dick, "Adaptive Chip-Package Thermal Analysis for Synthesis and Design," in *Proc. Design, Automation, and Test in Europe*, pp. 844–849, Mar. 2006.

[10] Y. Yang, Z. P. Gu, C. Zhu, R. P. Dick, and L. Shang, "ISAC: Integrated Space and Time Adaptive Chip-Package Thermal Analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Jan. 2007.

[11] C. Zhu, Z. Gu, L. Shang, R. P. Dick, and R. Knobel, "Towards an ultra-low-power architecture using single-electron tunneling transistors," in *Proc. Design Automation Conf.*, pp. 312–317, June 2007.

[12] Z. Gu, C. Zhu, L. Shang, and R. P. Dick, "Application-specific MPSoC reliability optimization," *IEEE Trans. VLSI Systems*. To appear.

[13] R. Kalla, B. Sinharoy, and J. M. Tendler, "IBM Power5 chip: a dual-core multithreaded processor," *IEEE Micro*, vol. 24, no. 2, pp. 40–47, 2004.

[14] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-way multithreaded SPARC processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, 2005.

[15] "AMD multi-core white paper." http://www.amd.com.

[16] "Intel multi-core processor architecture." http://www.intel.com.

[17] M. B. Taylor, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "Evaluation of the raw microprocessor: An exposed-wire-delay architecture for ILP and streams," in *Proc. Int. Symp. Computer Architecture*, June 2004.

[18] K. Sankaralingam, R. Nagarajan, H. Liu, J. Huh, C. K. Kim, D. Burger, S. W. Keckler, and C. R. Moore, "Exploiting ILP, TLP, and DLP using polymorphism in the TRIPS architecture," in *Proc. Int. Symp. Computer Architecture*, June 2003.

[19] K. Olukotun, B. Nayfeh, L. Hammond, K. Wilson, and K.-Y. Chang, "The case for a single-chip multiprocessor," in *Proc. the 7th Int. Symposium on Architectural Support for Programming Languages and Operating Systems*, Oct. 1996.

[20] S. J. Souri, K. Banerjee, A. Mehrotra, and K. C. Saraswat, "Multiple Si layer ICs: Motivation, performance analysis, and design implications," in *Proc. Design Automation Conf.*, pp. 213–220, June 2000.

[21] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICs: a novel chip design for improving deep-submicrometer interconnect performance and systems," *Proc. IEEE*, vol. 89, no. 5, 2001.

[22] S. Das, A. Fan, K.-N. Chen, and C. S. TanAnantha, "Technology, performance, and computer-aided design of three-dimensional integrated circuits," in *Proc. Int. Symp. Physical Design*, pp. 108–115, Apr. 2004.

[23] A. W. Topol, D. C. L. Tulipe, L. S. Jr., D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong, "Three-dimensional integrated circuits," *IBM J. Research and Development*, vol. 4, 2006.

[24] R. Camposano and W. Wolf, *High Level VLSI Synthesis*. Kluwer Academic Publishers, MA, 1991.

[25] D. C. Ku and G. D. Micheli, *High Level Synthesis of ASICs Under Timing and Synchronization Constraints*. Kluwer Academic Publishers, MA, 1992.

[26] D. Gajski, N. Dutt, A. Wu, and S. Lin, *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Publishers, MA, 1992.

[27] A. Raghunathan and N. K. Jha, "SCALP: An iterative-improvement-based low-power data path synthesis system," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 1260–1277, Nov. 1997.

[28] P. G. Paulin, J. P. Knight, and E. F. Girczyc, "HAL: A multi-paradigm approach to automatic data path synthesis," in *Proc. Design Automation Conf.*, pp. 263–270, June 1986.

[29] R. K. Gupta and G. De Micheli, "Hardware-software cosynthesis for digital systems," *IEEE Design & Test of Computers*, vol. 10, pp. 29–41, Sept. 1993.

[30] R. Mehra and J. Rabaey, "Behavioral level power estimation and exploration," in *Proc. Int. Wkshp. on Low Power Design*, pp. 197–202, Apr. 1994.

[31] A. Dasgupta and R. Karri, "Simultaneous scheduling and binding for power minimization during microarchitecture synthesis," in *Proc. Int. Symp. Low-Power Design*, Apr. 1994.

[32] L. Goodby, A. Orailoglu, and P. M. Chau, "Microarchitecture synthesis of performance-constrained, low-power VLSI designs," in *Proc. Int. Conf. Computer Design*, Oct. 1994.

[33] A. Raghunathan and N. K. Jha, "Behavioral synthesis for low power," in *Proc. Int. Conf. Computer Design*, pp. 318–322, Oct. 1994.

[34] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. Brodersen, "Optimizing power using transformations," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, pp. 12–31, Jan. 1995.

[35] R. S. Martin and J. P. Knight, "Power profiler: Optimizing ASICs power consumption at the behavioral level," in *Proc. Design Automation Conf.*, June 1995.

[36] J. M. Chang and M. Pedram, "Register allocation and binding for low power," in *Proc. Design Automation Conf.*, June 1995.

[37] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven behavioral synthesis for low-power VLSI systems," *IEEE Design Test*, vol. 12, no. 3, pp. 70–84, 1995.

[38] K. S. Khouri, G. Lakshminarayana, and N. K. Jha, "High-level synthesis of low power control-flow intensive circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 1715–1729, Dec. 1999.

[39] H. P. Peixoto and M. F. Jacome, "A new technique for estimating lower bounds on latency for high level synthesis," in *Proc. Great Lakes Symp. VLSI*, pp. 129–132, Mar. 2000.

[40] M. C. McFarland and T. J. Kowalski, "Incorporating bottom-up design into hardware synthesis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, pp. 938–950, Sept. 1990.

[41] D. W. Knapp, "Fasolt: A program for feedback-driven data-path optimization," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, pp. 677–695, June 1992.

[42] J. P. Weng and A. C. Parker, "3D scheduling: High-level synthesis with floorplanning," in *Proc. Design Automation Conf.*, June 1992.

[43] Y. M. Fang and D. F. Wong, "Simultaneous functional-unit binding and floorplanning," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994.

[44] M. Xu and F. J. Kurdahi, "Layout-driven RTL binding techniques for high-level synthesis using accurate estimators," *ACM Trans. Design Automation Electronic Systems*, vol. 2, pp. 312–343, Oct. 1997.

[45] W. E. Dougherty and D. E. Thomas, "Unifying behavioral synthesis and physical design," in *Proc. Design Automation Conf.*, June 2000.

[46] P. G. Paulin and J. P. Knight, "Scheduling and binding algorithms for high-level synthesis," in *Proc. Design Automation Conf.*, pp. 1–6, June 1989.

[47] C. A. Papachristou and H. Konuk, "A linear program driven scheduling and allocation method followed by an interconnect optimization algorithm," in *Proc. Design Automation Conf.*, June 1990.

[48] T. A. Ly, W. L. Elwood, and E. F. Girczyc, "A generalized interconnect model for data path synthesis," in *Proc. Design Automation Conf.*, June 1990.

[49] S. Tarafdar and M. Leeser, "The DT-model: High-level synthesis using data transfer," in *Proc. Design Automation Conf.*, June 1998.

[50] C. Jego, E. Casseau, and E. Martin, "Interconnect cost control during high-level synthesis," in *Proc. Design Circuits & Integration Systems Conf.*, Nov. 2000.

[51] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, 2001.

[52] P. Prabhakaran and P. Banerjee, "Simultaneous scheduling, binding and floorplanning high-level synthesis," in *Proc. Int. Conf. VLSI Design*, Jan. 1998.

[53] L. Zhong and N. K. Jha, "Interconnect-aware low power high-level synthesis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 336–351, Mar. 2005.

[54] A. Stammermann, D. Helms, M. Schulte, A. Schulz, and W. Nebel, "Binding, allocation and floorplanning in low power high-level synthesis," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003.

[55] O. Coudert, J. Cong, S. Malik, and M. Sarrafzadeh, "Incremental CAD," in *Proc. Int. Conf. Computer-Aided Design*, pp. 236–244, Nov. 2000.

[56] W. Choi and K. Bazargan, "Hierarchical global floorplacement using simulated annealing and network flow migration," in *Proc. Design, Automation & Test in Europe Conf.*, Mar. 2003.

[57] H. Zhou and J. Wang, "ACG–Adjacent constraint graph for general floorplans," in *Proc. Int. Conf. Computer Design*, Oct. 2004.

[58] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proc. Design Automation Conf.*, pp. 173–181, June 1982.

[59] L. Stockmeyer, "Optimal orientations of cells in slicing floorplan designs," *Information & Control*, vol. 57, pp. 91–101, May 1983.

[60] J. Wang and H. Zhou, "Interconnect estimation without packing via ACG floorplans," in *Proc. Asia & South Pacific Design Automation Conf.*, Jan. 2005.

[61] J. Wang, "Floorplanning by adjacent constrain graph and its applications," Master's thesis, Northwestern University, June 2005.

[62] "Independent JPEG group." www.ijp.org.

[63] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic, NY, 1990.

[64] "NCSU CBL high-level synthesis benchmark suite." www.cbl.ncsu.edu/benchmarks.

[65] S. Y. Kung, *VLSI Array Processors*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[66] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task Graphs for Free," in *Proc. Int. Wkshp. Hardware/Software Co-Design*, pp. 97–101, Mar. 1998.

[67] J. Cong and Z. Pan, "Interconnect performance estimation models for design planning," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 739–752, June 2001.

[68] J. Cong and M. Sarrafzadeh, "Incremental physical design," in *Proc. Int. Symp. Physical Design*, Apr. 2000.

[69] R. Mehra, L. M. Guerra, and J. M. Rabaey, "Low power architecture synthesis and impact of exploiting locality," *J. VLSI Signal Processing*, vol. 13, pp. 877–888, Aug. 1996.

[70] L. Zhong and N. K. Jha, "Interconnect-aware high-level synthesis for low power," in *Proc. Int. Conf. Computer-Aided Design*, pp. 110–117, Nov. 2002.

[71] Y. Cheng, P. Raha, C. Teng, E. Rosenbaum, and S. Kang, "ILLIADS-T: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 668–681, Aug. 1998.

[72] Z. Yu, D. Yergeau, R. Dutton, S. Nakagawa, and J. Deeney, "Fast placement-dependent full chip thermal simulation," in *Proc. Int. Symp. VLSI Tech., Systems, & Applications*, pp. 249–252, Apr. 2001.

[73] P. Li, L. T. Pileggi, M. Ashghi, and R. Chandra, "Efficient full-chip thermal modeling and analysis," in *Proc. Int. Conf. Computer-Aided Design*, pp. 319–326, Nov. 2004.

[74] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *Proc. Int. Symp. Microarchitecture*, pp. 294–305, Dec. 2002.

[75] X. Guo, D. Celo, P. Gunpudi, R. Khazaka, D. J. Walkey, T. Smy, and M. Nakhla, "The creation of compact thermal models of electronic components using model reduction," in *Proc. Semiconductor Thermal Measurement & Management Symp.*, pp. 104–110, Mar. 2004.

[76] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. Int. Symp. Computer Architecture*, pp. 2–13, June 2003.

[77] K. Banerjee, A. Mehrotra, A. Sanjiovanni-Vincentelli, and C. Hu, "On thermal effects in deep sub-micron VLSI interconnects," in *Proc. Design Automation Conf.*, pp. 885–891, June 1999.

[78] T.-Y. Chiang, K. Banerjee, and K. C. Saraswat, "Analytical thermal model for multilevel VLSI interconnects incorporating via effect," *IEEE Electron Device Ltrs.*, vol. 23, pp. 31–33, Jan. 2002.

[79] Z. Lu, W. Huang, J. Lach, M. Stan, and K. Skadron, "Interconnect lifetime prediction under dynamic stress for reliability-aware design," in *Proc. Int. Conf. Computer-Aided Design*, pp. 327–334, Nov. 2004.

[80] C. Tsai and S. Kang, "Cell-level placement for improving substrate thermal distribution," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, pp. 253–266, Feb. 2000.

[81] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *Proc. Int. Conf. Computer-Aided Design*, pp. 86–89, Nov. 2003.

[82] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *Proc. Int. Conf. Computer-Aided Design*, pp. 306–313, Nov. 2004.

[83] R. Mukherjee, S. Öğrenci Memik, and G. Memik, "Temperature-aware resource allocation and binding in high-level synthesis," in *Proc. Design Automation Conf.*, June 2005.

[84] Y. Cai, B. Liu, Q. Zhou, and X. Hong, "A thermal aware floorplanning algorithm supporting voltage island for low power soc design," *Integrated Circuit and System Design*, Aug. 2005.

[85] W.-L. Hung, G. Link, Y. Xie, N. Vijaykrishnan, N. Dhanwada, and J. Conner, "Temperature-aware voltage islands architecting in system-on-chip design," in *Proc. Int. Conf. Computer Design*, Oct. 2005.

[86] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, "Voltage island aware floorplanning for power and timing optimization," in *Proc. Int. Conf. Computer-Aided Design*, nov. 2006.

[87] R. L. Ching, E. F. Young, K. C. Leung, and C. Chu, "Post-placement voltage island generation," in *Proc. Int. Conf. Computer-Aided Design*, nov. 2006.

[88] Z. P. Gu, J. Wang, R. P. Dick, and H. Zhou, "Unified incremental physical-level and high-level synthesis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Sept. 2007.

[89] K. A. Bowman, B. L. Austin, J. C. Eble, X. Tang, and J. D. Meindl, "A physical alpha-power law MOSFET model," *IEEE J. Solid-State Circuits*, vol. 34, pp. 1410–1414, Oct. 1999.

[90] S. Ghiasi, E. Bozorgzadeh, P.-K. Huang, R. Jafari, and M. Sarrafzadeh, "A unified theory of timing budget management," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 2364–2375, Nov. 2006.

[91] L. Shang, L.-S. Peh, A. Kumar, and N. K. Jha, "Thermal modeling, characterization and management of on-chip networks," in *Proc. Int. Symp. Microarchitecture*, pp. 67–80, Dec. 2004.

[92] "COMSOL Multiphysics." http://www.comsol.com/products/multiphysics.

[93] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Exploiting structural duplication for lifetime reliability enhancement," in *Proc. Int. Symp. Computer Architecture*, pp. 520–531, June 2005.

[94] B. Dave, G. Lakshminarayana, and N. K. Jha, "COSYN: Hardware-software co-synthesis of embedded systems," in *Proc. Design Automation Conf.*, pp. 703–708, June 1997.

[95] Y. Xie, L. Lu, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Reliability-aware co-synthesis for embedded systems," in *Proc. Int. Conf. Application-Specific Systems, Architectures, and Processors*, Sept. 2004.

[96] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear programming based techniques for synthesis of network-on-chip architectures," in *Proc. Int. Conf. Computer Design*, pp. 422–429, Oct. 2004.

[97] U. Y. Ogras and R. Mărculescu, "Energy- and performance- driven NoC communication architectures synthesis using a decomposition approach," in *Proc. Design, Automation & Test in Europe Conf.*, pp. 352–357, Mar. 2005.

[98] R. P. Dick and N. K. Jha, "MOCSYN: Multiobjective Core-Based Single-Chip System Synthesis," in *Proc. Design, Automation & Test in Europe Conf.*, pp. 263–270, Mar. 1999.

[99] Y. Yang, C. Zhu, Z. P. Gu, L. Shang, and R. P. Dick, "Adaptive multi-domain thermal modeling and analysis for integrated circuit synthesis and design," in *Proc. Int. Conf. Computer-Aided Design*, pp. 575–582, Nov. 2006.

[100] R. P. Dick, "E3S: The embedded system synthesis benchmarks suite." E3S link at http://robertdick.org/tools.html.

[101] S. W. Mahfoud and D. E. Goldberg, "Parallel recombinative simulated annealing: A genetic algorithm," *Parallel Computing*, vol. 21, pp. 1–28, Jan. 1995.

[102] R. P. Dick, *Multiobjective Synthesis of Low-Power Real-Time Distributed Embedded Systems*. PhD thesis, Dept. of Electrical Engineering, Princeton University, July 2002.

[103] R. P. Dick and N. K. Jha, "MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Co-Synthesis of Distributed Embedded Systems," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 920–935, Oct. 1998.

[104] Joint Electron Device Engineering Council, "Failure mechanisms and models for semiconductor devices," in *JEDEC Publication JEP 122-B*, Aug. 2003.

[105] L. Ting, J. May, W. Hunter, and J. McPherson, "AC electromigration characterization and modeling of multilayered interconnections," in *Proc. Int. Reliability Physics Symp.*, pp. 311–316, Mar. 1993.

[106] C. Dunn and J. McPherson, "Temperature-cycling acceleration factors for aluminum metallization failure in VLSI applications," in *Proc. Int. Reliability Physics Symp.*, pp. 252–255, Mar. 1990.

[107] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, 1989.

[108] A.-R. Chowdhury and P. Banerjee, "A new error analysis based method for tolerance computation for algorithm-based checks," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 45, pp. 238–243, Feb. 1996.

[109] A. Mishra and P. Banerjee, "An algorithm-based error detection scheme for the multigrid method," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 52, pp. 1089–1099, Sept. 2003.

[110] Y. Zhang, R. P. Dick, and K. Chakrabarty, "Energy-aware deterministic fault tolerance in distributed real-time embedded systems," in *Proc. Design Automation Conf.*, pp. 550–555, June 2004.

[111] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. Int. Conf. Genetic Algorithms*, pp. 416–423, July 1993.

[112] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, England, 1990.

[113] "Embedded microprocessor benchmark consortium." http://www.eembc.org.

[114] T. Sakurai, "A JSSC classic paper: The simple model of CMOS drain current," *IEEE Solid State Circuits Society Quarterly Newsletter*, pp. 4–5, Oct. 2004.

[115] M. S. Dresselhaus, G. Dresselhaus, and P. Avouris, *Carbon Nanotubes*. Springer-Verlag, Germany, Feb. 2001.

[116] Y. Huang, X. Duan, Y. Cui, L. J. Lauhon, K.-H. Kim, and C. M. Lieber, "Logic gates and computation from assembled nanowire building blocks," *Nature*, vol. 294, pp. 1313–1317, Nov. 2001.

[117] K. K. Likharev, "Single-electron devices and their applications," *Proc. IEEE*, vol. 87, pp. 606–632, Apr. 1999.

[118] D. V. Averin and K. K. Likharev, "Coulomb blockade of tunneling and coherent oscillations in small tunnel junctions," *J. Low Temperature Physics*, vol. 62, pp. 345–372, Feb. 1986.

[119] T. A. Fulton and G. J. Dolan, "Observation of single-electron charging effects in small tunnel junctions," *Physics Review Ltrs.*, vol. 59, pp. 109–112, July 1987.

[120] M. H. Devoret and R. J. Schoelkopf, "Amplifiying quantum signals with the single-electron transistor," *Nature*, vol. 406, pp. 1039–1046, Aug. 2000.

[121] Y. Nakamura, C. D. Chen, and J. S. Tsai, "100-K operation of Al-based single-electron transistors," *Japan Journal Applied Physics*, vol. 35, pp. 1465–1467, Nov. 1996.

[122] D. Klein, R. Roth, A. K. L. Lim, A. P. Alivisatos, and P. McEuen, "A single-electron trnaistor made from a cadmium selenide nanocrystal," *Nature*, vol. 389, pp. 699–701, Oct. 1997.

[123] X. Tang, X. Baie, V. Bayot, F. V. de Wiele, and J. P. Colinge, "An SOI single-electron transistor," in *Proc. Silicon-on-Insulator Conf.*, pp. 46–47, Oct. 1999.

[124] M. Ahlskog, R. Tarkiainen, L. Roschier, and P. Hakonen, "Single-electron transistor made of two crossing multiwalled carbon nanotubes and its noise properties," *Applied Physics Ltrs.*, vol. 77, pp. 4037–4039, Dec. 2000.

[125] K. Matsumoto, M. Ishii, K. Segawa, and Y. Oka, "Room temperature operation of a single electron transistor made by the scanning tunneling microscope nanooxidation process for the TiO_x/Ti system," *Applied Physics Ltrs.*, vol. 68, pp. 34–36, Jan. 1996.

[126] E. S. Soldatov, V. V. Kahanin, A. S. Trifononv, S. P. Gubin, V. V. Kolesov, D. E. Presnov, S. A. Yakovenko, G. B. Khomutov, and A. N. Korotkov, "Room temperature molecular single-electron transistor," *JETP Ltrs.*, vol. 64, pp. 556–558, Oct. 1996.

[127] J.-I. Shirakashi, K. Matsumoto, N. Miura, and M. Konagai, "Single-electron charging effects in Nb/Nb oxide-based single-electron transistors at room temperature," *Applied Physics Ltrs.*, vol. 72, pp. 1893–1895, Apr. 1998.

[128] Y. A. Pashkin, Y. Nakamura, and J. S. Tsai, "Room-temperature Al single-electron transistor made by electron-beam lithography," *Applied Physics Ltrs.*, vol. 76, pp. 2256–2258, Apr. 2000.

[129] J. R. Tucker, "Complementary digital logic based on the Coulomb blockade," *J. Applied Physics*, vol. 72, no. 99, pp. 4399–4413, 1992.

[130] R. H. Chen, A. N. Korotkov, and K. K. Likharev, "Single-electron transistor logic," *Applied Physics Ltrs.*, vol. 68, pp. 1954–1956, Apr. 1996.

[131] K. Uchida, J. Koga, R. Ohba, and A. Toriumi, "Programmable single-electron transistor logic for future low-power intelligent LSI: proposal and room-temperature operation," *IEEE Trans. Electron Devices*, vol. 50, pp. 1623–1630, July 2003.

[132] F. Nakajima, Y. Miyoshi, J. Motohisa, and T. Fukui, "Single-electron AND/NAND logic circuits based on a self-organized dot network," *Applied Physics Ltrs.*, vol. 83, Sept. 2003.

[133] Y.-K. Cho and Y.-H. Jeong, "Single-electron pass-transistor logic with multiple tunnel junctions and its hybrid circuit with MOSFETs," *ETRI J.*, vol. 26, pp. 669–672, Dec. 2004.

[134] K. Yano, T. Ishii, T. Hashimoto, T. Kobayashi, F. Murai, and K. Seki, "Room-temperature single-electron memory," *IEEE Trans. Electron Devices*, vol. 41, pp. 1628–1638, Sept. 1994.

[135] C. Wasshuber, H. Kosina, and S. Selberherr, "A comparative study of single electron memories," *IEEE Trans. Electron Devices*, vol. 45, pp. 2365–2371, Nov. 1998.

[136] K. K. Yadavalli, A. O. Orlov, G. L. Snider, and A. N. Korotkov, "Single electron memory devices: toward background charge insensitive operation," *J. Vacuum Science Technology B Microelectronics and Nanometer Structures*, vol. 21, pp. 2860–2864, 2003.

[137] C. Wasshuber, H. Kosina, and S. Selberherr, "A single-electron device and circuit simulator," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 937–944, Sept. 1997.

[138] R. H. Chen, "MOSES: a general Monte Carlo simulator for single-electron circuits," *Meeting Abstracts, The Electrochemical Society*, vol. 96, p. 576, Oct. 1996.

[139] K. Uchida, K. Matsuzawa, J. Koga, R. Ohba, S. ichi Takagi, and A. Toriumi, "Analytical single-electron transistor (SET) model for design and analysis of realistic set circuits," *Japanese. J. Applied Physics*, vol. 39, pp. 2321–2324, Apr. 2000.

[140] H. Inokawa and Y. Takahashi, "A compact analytical model for asymmetric single-electron tunneling transistors," *IEEE Trans. Electron Devices*, vol. 50, pp. 455–461, Feb. 2003.

[141] S. Mahapatra, V. Vaish, C. Wasshuber, and K. Banerjee, "Analytical modelling of single electron transistor (SET) for hybrid CMOS-SET analog IC design," *IEEE Trans. Electron Devices*, vol. 51, pp. 1772–1782, June 2004.

[142] J. R. Heath and M. A. Ratner, "Molecular electronics," *Physics Today*, vol. 56, pp. 43–49, May 2003.

[143] A. K. Geim and K. S. Novoselov, "The rise of graphene," *Nature Materials*, vol. 6, pp. 183–191, Mar. 2007.

[144] S. Vanapalli, M. Lewis, Z. Gan, and R. Radebaugh, "120 Hz pulse tube cryocooler for fast cooldown to 50 K," *Applied Physics Letters*, vol. 90, Feb. 2007.

[145] D. K. Ferry and S. M. Goodnick, *Transport in Nanostructures*. Cambridge University Press, 1997.

[146] Y. Ono, Y. Takahashi, K. Yamazaki, M. Nagase, H. Namatsu, K. Kurihara, and K. Murase, "Si complementary single-electron inverter," *IEDM Technology Dig.*, pp. 367–370, 1999.

[147] C. P. Heij, P. Hadley, and J. E. Mooij, "Single-electron inverter," *Applied Physics Ltrs.*, vol. 78, pp. 1140–1142, 2001.

[148] H. Wolf, F. J. Ahlers, J. Niemeyer, H. Scherer, T. Weimann, A. B. Zorin, V. A. Krupenin, S. V. Lotkhov, and D. E. Presnov, "Investigation of the offset charge noise in single electron tunneling devices," *Trans. on Instrumentation and Measurement*, vol. 46, Apr. 1997.

[149] M. Furlan and S. V. Lotkhov, "Electrometry on charge traps with a single-electron transistor," *Physics Rev. B*, vol. 67, p. 205313, 2003.

[150] V. A. Krupenin, D. Presnov, A. Zorin, and J. Niemeyer, "Aluminum single electron transistors with islands isolated from a substrate," *J. of Low Temperature Physics*, vol. 118, Dec. 1999.

[151] N. M. Zimmerman, W. H. Huber, A. Fujiwara, and Y. Takahashi, "Excellent charge offset stability in Si-based SET transistors," in *Proc. Precision Electromagnetic Measurements*, pp. 124–125, Nov. 2002.

[152] N. S. Zimmerman, W. H. Huber, A. Fujiwara, and Y. Takahashi, "Excellent charge offset stability in a Si-based single-electron tunneling transistor," *Applied Physics Ltrs.*, vol. 79, pp. 3186–3190, 2002.

[153] Y. S. Yu, S. W. Hwang, and D. Ahn, "Transient modelling of single-electron transistors for efficient circuit simulation by SPICE," *Electronics Ltrs.*, vol. 152, pp. 691–696, Dec. 2005.

[154] M. Kirihara, K. Nakazato, and M. Wagner, "Hybrid circuit simulator including a model for single electron tunneling devices," *Japanese J. of Applied Physics*, vol. 38, Apr. 1999.

[155] J. M. Rabaey, *Digital Integrated Circuits*. Prentice-Hall, NJ, 1998.

[156] A. DeHon, "Array-based architecture for FET-based nanoscale electronicss," *IEEE Trans. Nanotechnology*, vol. 2, Mar. 2003.

[157] S. C. Goldstein and M. Budiu, "Nanofabrics: spatial computing using molecular electronics," in *Proc. Int. Symp. Computer Architecture*, pp. 178–189, June 2001.

[158] R. I. Bahar, J. Mundy, and J. Chen, "A probabilistic-based design methodology for nanoscale computation," in *Proc. Int. Conf. Computer-Aided Design*, pp. 480–486, Nov. 2003.

[159] K. R. Brown, L. Sun, and B. E. Kane, "Electric-field-dependent spectroscopy of charge motion using a single-electron transistor," *Applied Physics Ltrs.*, vol. 88, 2006 May.

[160] "Xilinx XPower." http://www.xilinx.com.

[161] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," in *Proc. Int. Symp. Quality of Electronic Design*, pp. 585–590, Mar. 2006.

[162] S. Roundy, P. K. Wright, and J. Rabaey, "A study of low level vibrations as a power source for wireless sensor nodes," *Computer Communications*, vol. 26, Oct. 2003.

[163] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Lyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-tile 1.28TFLOPS networks-on-chip in 65nm CMOS," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2007.

[164] B. Black, M. M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb, "Die stacking (3D) microarchitecture," in *Proc. Int. Symp. Microarchitecture*, pp. 469–479, Dec. 2006.

[165] Samsung. http://www.samsung.com/.

[166] Tezzaron. http://www.tezzaron.com/technology/FaStack.htm.

[167] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir, "Design and management of 3D chip multiprocessors using network-in-memory," in *Proc. Int. Symp. Computer Architecture*, June 2006.

[168] T. Kgil, S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Reinhardt, and K. Flautner, "PicoServer: using 3D stacking technology to enable a compact energy efficient chip multiprocessor," in *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Oct. 2006.

[169] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das, "A novel dimensionally-decomposed router for on-chip communication in 3D architectures," in *Proc. Int. Symp. Computer Architecture*, June 2007.

[170] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proc. Int. Symp. High-Performance Computer Architecture*, Jan. 2001.

[171] M. D. Powell, M. Gomaa, and T. N. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system," in *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Nov. 2004.

[172] J.McGregor, "x86 power and thermal management," in *Microprocessor Report*, Dec. 2004.

[173] Y. Li, D. Brooks, Z. Hu, and K. Skadron, "Performance, energy, and thermal considerations for SMT and CMP architectures," in *Proc. Int. Symp. Computer Architecture*, Feb. 2005.

[174] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 simulator: Modeling networked systems," *Proc. Int. Symp. Microarchitecture*, vol. 26, no. 4, pp. 52–60, 2006.

[175] M. Healy, M. Vittes, M. Ekpanyapong, C. Ballapuram, S. K. Lim, H.-H. S. Lee, and G. H. Loh, "Multi-objective microarchitectural floorplanning for 2d and 3d ics," *TCAD*, vol. 26, pp. 38–52, Jan. 2007.

[176] Y. Tsai, Y. Xie, N. Vijaykrishnan, and M. J.Irwin, "Three-dimensional cache design exploration using 3DCacti," in *Proc. Int. Conf. Computer Design*, pp. 519–524, Oct. 2005.

[177] K. Puttaswamy and G. H. Loh, "Thermal analysis of a 3d die-stacked high-performance microprocessor," in *Proc. Great Lakes Symp. VLSI*, pp. 19–24, May 2006.

[178] K. Puttaswamy and G. H. Loh, "Thermal herding: Microarchitecture techniques for controlling hotspots in high-performance 3d-integrated processors," in *Proc. Int. Symp. High-Performance Computer Architecture*, pp. 193–204, Feb. 2007.

[179] G. L. Loi, B. Agrawal, N. Srivastava, S.-C. Lin, T. Sherwood, and K. Banerjee, "A thermally-aware performance analysis of vertically integrated (3-d) processor-memory hierarchy," in *Proc. Design Automation Conf.*, pp. 991–996, July 2006.

[180] G. M. Link and N. Vijaykrishnan, "Thermal trends in emerging technologies," in *Proc. Int. Symp. Quality of Electronic Design*, pp. 625–632, Mar. 2006.

[181] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotLeakage: A temperature-aware model of subthreshold and gate leakage for architects," tech. rep., Univ. of Virginia, May 2003. CS-2003-05.

[182] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. Computer Architecture*, pp. 83–94, June 2000.

[183] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Massubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, "The design and implementation of a frist-generation CELL processor," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2007.

[184] R. Sprunt, "Pentium 4 performance-monitoring features," *IEEE Micro*, vol. 22, no. 4, pp. 72–82, 2002.

[185] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proc. Int. Symp. Microarchitecture*, Dec. 2003.

[186] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha, "HybDTM: a coordinated hardware-software approach for dynamic thermal management," in *Proc. Design Automation Conf.*, pp. 548–553, July 2006.

[187] D. Tarjan, S. Thoziyoor, and N. P. Jouppi, "CACTI 4.0," tech. rep., HP Laboratories, June 2006.

[188] E. C. Samson, S. V. Machiroutu, J.-Y. Chang, I. Santos, J. Hermerding, A. Dani, R. Prasher, and D. W. Song, "Interface material selection and a thermal management technique in second-generation platforms built on Intel Centrino mobile technology," *Intel Technology J.*, vol. 09, pp. 75–86, Feb. 2005.

[189] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur, "Thermal performance challenges from silicon to systems," *Intel Technology J.*, vol. 04, Aug. 2000.

[190] A. Phansalkar, A. Joshi, L. Eeckhout, and L. K. John, "Measuring program similarity: Experiments with SPEC CPU benchmark suites," in *Proc. Int. Symp. on Performance Analysis of Systems and Software*, Mar. 2005.

[191] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The impact of technology scaling on lifetime reliability," in *Proc. International Conf. Dependable Systems and Networks*, pp. 177–186, June 2004.

[192] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal RC-modeling for accurate and localized dynamic thermal management," in *Proc. Int. Symp. High-Performance Computer Architecture*, pp. 17–28, Feb. 2001.

[193] Y. Li, B. Leez, D. Brooksz, Z. Huyy, and K. Skadron, "CMP design space exploration subject to physical constraints," in *Proc. Int. Symp. High-Performance Computer Architecture*, Feb. 2006.

[194] S. Heo, K. Barr, and K. Asanovic, "Reducing power density through activity migration," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 217–222, Aug. 2003.

[195] S. Park, W. Jiang, Y. Zhou, and S. Adve, "Managing energy-performance tradeoffs for multi-threaded applications," in *Proc. Int. Conf. on Measurement and Modeling of Computer Systems*, June 2007.

[196] J. Cong, Y. Fan, G. Han, X. Yang, and Z. Zhang, "Architecture and synthesis for on-chip multicycle communication," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, pp. 550–514, Apr. 2004.

[197] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, "Full chip leakage estimation considering power supply and temperature variations," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 78–83, Aug. 2003.

[198] K.-Y. Chao and D. F. Wong, "Thermal placement for high-performance multichip modules," in *Proc. Int. Conf. Computer Design*, pp. 218–223, Sept. 1995.

[199] C. C. N. Chu and D. F. Wong, "A matrix synthesis approach to thermal placement," in *Proc. Int. Symp. Physical Design*, pp. 163–168, Apr. 1997.

[200] G. Chen and S. Sapatnekar, "Partition-driven standard cell thermal placement," in *Proc. Int. Symp. Physical Design*, pp. 75–80, Apr. 2003.

Vita

Zhenyu Gu was born in Shanghai, China on July 15, 1978, the only son of Xiong Gu and Kunfang Zhang. He received his Bachelor's degree in electrical engineering from Fudan University in 2000, and Master's degree from Fudan University's ASIC and System State-Key Lab in 2003. His research interests include chip-level multiprocessor synthesis, unified behavioral and physical-level synthesis, thermal and reliability aware design and analysis, and design methodologies for VLSI systems and embedded systems.

He is the recipient of Outstanding Graduate Student Award of Shanghai (2003), Shanghai Applied Materials Fund Graduate Student Fellowship (2003), Alcatel Microelectronics Fellowship (2002), Guang Hua Fellowship (2001), 2nd Prize of the 4th National Undergraduate Student Electronics Design Contest (2000), Alcatel Fellowship (1999), Motorola Fellowship (1998), People's Fellowship (1997), and Freshmen Fellowship of Fudan University (1996).

He has four peer-review publications in IEEE conferences in the areas of design automation and VLSI design. His ASPDAC paper was one of eight best paper award candidates among 135 technical papers and his DAC paper was nominated for best paper award. When he was pursuing his doctoral degree, he was employed as a teaching and research assistant in Electrical Engineering and Computer Science Department of Northwestern University. He was also with Shanghai-Fudan Microelectronic Corp., Shanghai, China in the summers of 2000 and 2001.