Interconnect Design Techniques for Multicore and 3D Integrated Circuits

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Pingqiang Zhou

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Sachin S. Sapatnekar

August, 2012

Acknowledgements

First of all, I would like to express my deepest appreciation and gratitude to my advisor, Professor Sachin Sapatnekar, for his guidance, encouragement and support throughout the five years of my doctoral study at University of Minnesota, Twin Cities. He turned my dream of earning a PhD and becoming a rigorous researcher into a reality. It has been my great pleasure and an honor working with him.

I am deeply grateful to Professor Antonia Zhai and Professor Chris Kim, who have provided the precious help and guidance to my research and contributed to a significant portion of this thesis work.

Thanks to my PhD committee members, Professor Antonia Zhai, Professor Chris Kim and Professor Keshab Parhi, for reviewing my thesis and giving valuable feedbacks.

I own many thanks to colleagues in the VEDA Lab and the University of Minnesota for their help and many meaningful discussions: Baktash Boghrati, Sanjay Kumar, Qunzeng Liu, Ping-Hung Yuh, Jianxin Fang, Yaoguang Wei, Xianghong Liu, Saket Gupta, Chi Xu, Vivek Mishra, Sravan Marella, Jieming Yin, Dong Jiao, Bongjin Kim, Wonho Choi, Xiaofei Wang, Sudhir Kudva, Weikang Qian, and many others.

I am grateful to National Science Foundation and Semiconductor Research Cooperation for funding my research, and to the IBM T. J. Watson Research Lab for providing me the opportunity to work as an Intern.

Finally, I would like to thank my parents for their unconditional love, support and encouragement throughout my life. My special thanks go to my wife, Chao Liang, for her ever-present love, encouragement and support throughout these years. I cannot imagine going through this journey without her by my side.

Abstract

Over the past 40 years, the semiconductor industry has witnessed the exponential growth trend in system complexity as predicted by Moore's law, facilitated by continuously shrinking transistor and wire dimensions. *Three dimensional* (3D) circuit technologies, with multiple tiers of active devices stacked above each other, are a key approach to achieve increasing levels of integration and performance in the future. Concomitant with exponentially reducing device dimensions, designers face new challenges in maximizing computation while remaining with a stringent power envelope. Over the last decade, *multicore processors* have emerged as a potential solution to address some of these problems by integrating multiple smaller and more energy efficient cores in order to replace a single, larger core. These cores must communicate through an efficient on-chip interconnection network, by ideas such as *networks-on-chips* (NoCs), and NoC design is vital to both performance and power. This thesis presents solutions to the challenges in on-chip interconnect, more specifically, the on-chip communication and power delivery networks of 3D and multicore chips.

The first part of this thesis focuses on developing techniques for designing efficient and high-performance NoC architecture for 3D and multicore chips. Depending on the nature of the application, the multicore system may be either a System-on-Chip (SoC), which executes a relatively well-characterized workload, or a Chip multiprocessor (CMP), which is a general purpose processor that should be capable of handling a variety of workloads. **For SoCs**, this thesis presents an efficient algorithm to synthesize application-specific NoC architectures in 3D environment. We demonstrate that this method finds greatly improved solutions compared to a baseline algorithm reflecting prior work. We also study the impact of various factors on the network performance in 3D NoCs, including the through-silicon via (TSV) count and the number of 3D tiers. **For CMPs**, we observe that voltage and frequency scaling (VFS) for NoC can potentially reduce energy consumption, but the associated increase in latency and degradation in throughput limits its deployment. Therefore, we propose flexible-pipeline routers that reconfigure pipeline stages upon VFS, so that latency through such routers remains constant. With minimal hardware overhead, the deployment of such routers allows us to reduce network frequency and save network energy, without significant performance degradation.

The second part of this thesis is concerned with the design and optimization of power delivery network for 3D and multicore chips. **First**, we propose a novel paradigm where we exploit a new type of capacitor, the metal-insulator-metal (MIM) capacitor, together with the traditional CMOS decaps, to optimize the power supply noise in 3D chips. Experimental results show that power grid noise can be more effectively optimized after the introduction of MIM decaps, with lower leakage power and little increase in the routing congestion, as compared to a solution using CMOS decaps only. **Second**, we explore the design and optimization of on-chip switched-capacitor (SC) DC-DC converters for multicore processors. On one hand, with an accurate power grid simulator, we find that distributed design of SC converters can reduce the IR drop significantly compared to the lumped design, with improved supply voltage. On the other hand, the efficiency of the power delivery system using SC converters is a major concern, but this has not been addressed at the system level in prior research, we develop models for the efficiency of such a system as a function of size and layout of the SC converters, and proposes an approach to optimize the size and layout of the SC converter to minimize power loss. The efficiency of these techniques is demonstrated on both homogenous and heterogenous multicore chips.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the past 40 years, the semiconductor industry has been driven by Moore's law, which has correctly predicted that the number of transistors integrated on a chip will double every 18-24 months, resulting in exponential growth in chip complexity. This trend was originally forecast in 1965 based on only five data points, the largest of which corresponded to just 64 on-chip transistors. Amazingly, it has remained an accurate predictor from then until today's 3 billion transistor designs [3]. In part, this is due to the semiconductor industry's efforts to make Moore's "law" a self-fulfilling prophecy, through strategic plans such as those outlined in the International Technology Roadmap for Semiconductors [4], to drive the industry and the overall supply chain to achieve and sustain this impressive growth. On the demand side, this growth has been spurred on by the tremendous appetite for newer, faster, cheaper, and more mobile chips that have revolutionized our way of life, making a pervasive imprint across areas such as scientific computing, wireless communication, the internet, electronic entertainment, digital photography and videography, healthcare, security, and banking.

There are three significant impediments to the continuation of this trend. **First**, Moore's law has been facilitated by continuously shrinking transistor and wire dimensions, so that more devices can be fabricated within the same silicon area. However, these feature sizes are now down to tens of nanometers, where the cost of manufacturing is high. This has motivated the semiconductor industry to consider other avenues for increasing on-chip integration: of these, *three-dimensional (3D) integration* is fast emerging as a viable option for continuing the exponential trend. Today's integrated

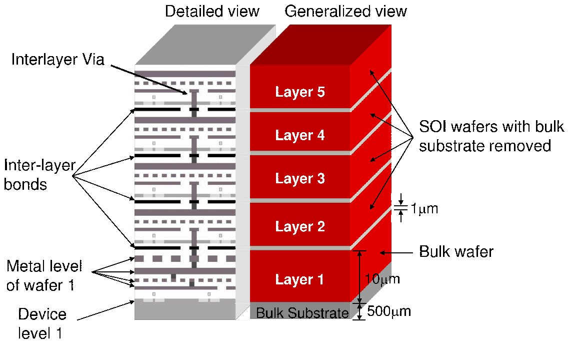


Figure 1.1: 3D integrated circuit.

circuits (ICs) are "2D" and place all devices in a single layer – in contrast, a 3D IC (Figure 1.1) stacks multiple layers of active devices above each other, providing the potential to increase transistor packing density and reduce chip area significantly [5]. **Second**, although Moore's law makes more devices available on a chip, running too many of them dissipates unacceptably high power and generates excessive heat. These limitations mean that a smaller fraction of all devices can remain on at a given time, and innovative power delivery and thermal management methods are essential. As one part of the solution, single-core processors have made way for multicore processors, which enable better power and thermal management. **Third**, as more devices have been placed on a chip, there is a need for greater communication between the devices. Conventional paradigms that use dedicated wires or buses do not scale well with system sizes, and novel ideas such as *networks-on-chip (NoCs)* [6] (Figure 1.2) are gaining traction for future on-chip communication architectures, particularly multicores.

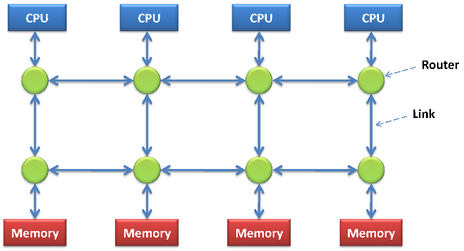


Figure 1.2: Schematic of an NoC architecture with routers and links.

My thesis is motivated by these three challenges and relates to optimizing interconnects for 3D and multicore chips, which is widely accepted [4] as the major performance bottleneck in future designs. My thesis work is to automate the design and optimization of interconnect, more specifically, the communication network and power delivery network, in 3D and multicore chips. The content of this dissertation can be classified into two broad chapters:

1.1 Global Communication in 3D and Multicore Chips

NoCs are a very new on-chip communication scheme and send out messages in "packets" (similar to message routing over the Internet, but over more severely constrained on-chip networks). NoCs are emerging as a scalable framework for the on-chip communication infrastructure of future densely-integrated designs, but significant hurdles (or as I see them, research opporutnities) remain before their potential can be realized. Depending on the nature of the application, the design may be either a System-on-Chip (SoC), which executes a relatively well-characterized workload, or a Chip Multiprocessor (CMP), which is a general purpose processor that should be capable of handling a variety of workloads. Simple types of NoCs have been used in experimental proof-of-concept systems that prototype future multicore processors [7], but a number of key issues remain unresolved. NoC design is vital to both performance and power, and my work is to build automated techniques for designing efficient, high-performance NoCs, both for SoCs and for CMPs.

1.1.1 Designing Application-specific NoC Architectures for 3D SoCs

3D technology enables the design of more complex and more highly interconnected systems: in this context, NoCs promise major benefits, but impose new constraints and limitations. This thesis presents a novel technique in Chapter 2 that simultaneously determines the chip layout (floorplan) and designs an application-specific 3D NoC architectures for custom SoCs. We have employed a stochastic flow allocation method, Simulated Allocation (SAL) [8], to route the traffic flows and build the topology for the application. The technique has a built-in thermal analyzer that ensures that the resulting solution avoids creating thermal "hot spots" that could degrade circuit performance and reliability. We demonstrate that this method finds greatly improved solutions compared to a baseline algorithm reflecting prior work. To evaluate the SAL method, we compare its performance with the widely-used simulated annealing (SA) method, and show that SAL is much faster than SA for this application while providing solutions of very similar quality. We also study the impact of various factors on the network performance in 3D NoCs, including the TSV count and the number of 3D tiers.

1.1.2 Optimization of NoCs for CMPs

For CMPs, we observe that they show intermittent or "bursty" traffic patterns (see the sudden peaks in Figure 1.3), leading us to conclude that methods that adaptively scale the voltage and frequency of the network (providing it just as much performance as necessary and saving power while doing so) can be used very effectively to potentially reduce NoC energy consumption.

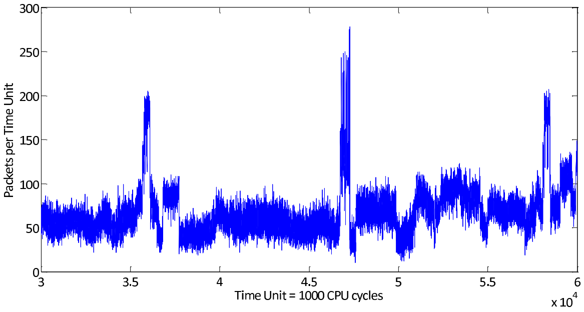


Figure 1.3: Observed traffic on an NoC link.

This thesis has proposed the notion of flexible pipeline routers in Chapter 3, where pipeline stages are reconfigured upon frequency scaling. For example, when the clock frequency is scaled down, our method decreases the number of pipeline stages, thus decreasing the latency through the system; this is supplemented with intelligent techniques that rebalance the logic between pipeline stages. Using such methods, our experiments show that, over a large class of applications that are only sensitive to NoC latency, deploying flexible-pipeline routers allows for *massive energy savings* with *little performance penalty* , as opposed to traditional fixed-pipeline routers.

1.2 Power Delivery in 3D and Multicore Chips

The power network distributes the power supply all over a chip, just as the terrestrial electricity grid sends out electricity for use in homes, offices, and factories – but the on-chip grid operates at smaller geometries and much higher power densities. Reliable power delivery is being recognized as a major challenge in 3D multicore processors, due to less voltage headroom by technology scaling, increased current density, and large switching transients between cores [9]. This thesis develops integrated design and CAD solutions for innovative on-chip power delivery techniques, specifically targeted for 3D and multicore processors.

1.2.1 Power Supply Network Optimization in 3D Circuits

A widely employed method for controlling supply voltage levels, to ensure correct chip performance, involves the use of decoupling capacitors (decaps) – deliberately-inserted capacitors whose task are to slow down transients in the supply network and keep voltage levels stable. Traditional thin-oxide decaps are becoming increasingly lossy with technology scaling: cumulatively for a circuit, their leakage current losses could lead to substantial wasted power and inefficiencies. In Chapter 4, we have proposed a novel paradigm where we exploit a new type of capacitor, the metal-insulator-metal (MIM) capacitor, which can be built economically, within conventional mass-produced fabrication technologies. MIM decaps dissipate near-zero leakage power, but unlike conventional decaps, they create obstacles in the circuit since wires cannot be routed through them.

Our work presents a best-of-both-worlds approach for decap allocation, using both conventional CMOS decaps and MIM decaps, quantifying the cost and benefit of each technology. We formulate the decap budgeting problem, using both CMOS and MIM decaps, as a Linear Programming (LP) problem, and propose an efficient congestion-aware algorithm to optimize the power supply noise. Experimental results demonstrate that the use of CMOS decaps alone is insufficient to overcome the violations; the use of MIM decaps results in high levels of congestion; and the optimal mix of the two is the *best solution* that meets both congestion and noise constraints, with low leakage.

1.2.2 Design and Optimization of On-Chip Power Delivery Network For Multicore Platforms

This thesis continues to devise techniques for building and optimizing the power delivery network for CMPs, using techniques that go beyond the decap optimization explored earlier. A very promising direction is related to a recent technical advance that makes it possible to integrate on-chip voltage regulator structures (Figure 1.4), which are very effective in maintaining supply voltage levels.

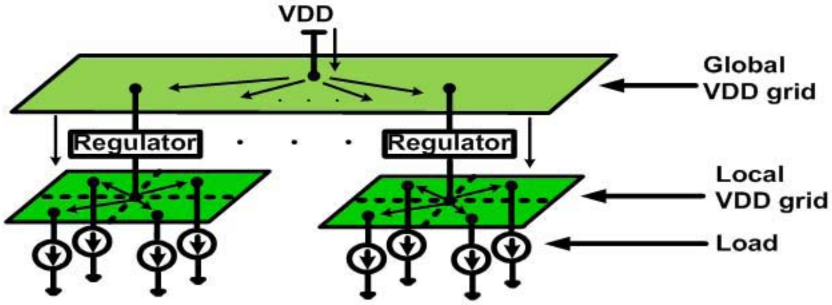


Figure 1.4: On-chip voltage regulators.

In collaboration with Professor Chris Kim's VLSI design group, we have explored the design of on-chip switched-capacitor (SC) DC-DC converters in the context of multicore processors. In Chapter 5, we explore the design of on-chip SC converters, using an accurate power grid simulator. Results show that distributed design of SC converters can reduce the IR drop significantly compared to the lumped design, with improved supply voltage. We also demonstrate the usage of SC converters for multi-domain power supply. The efficiency of the power delivery system using SC converters is a major concern, but this has not been addressed at the system level in prior research. In Chapter 6, we has developed CAD support for determining the optimal number and topologies of these voltage regulators under multicore workloads. We develop models for the efficiency of the power delivery system as a function of size and layout of the SC converters, and proposes an approach to optimize the size and layout of the SC converter to minimize power loss. The efficiency of these techniques is demonstrated on both homogenous and heterogenous multicore chips.

Chapter 2

Application-specific NoC design and optimization for SoCs

As presented in Chapter 1, three-dimensional (3D) silicon integration technologies have provided new opportunities for Network-on-Chip (NoC) architecture design in Systems-on-Chip (SoCs). In this chapter, we consider the application-specific NoC architecture design problem in a 3D environment for SoCs applications, which have static or semi-static traffic characteristics in the network. Dynamic traffic behaviors are observed in the many-core processors, and we study such applications in Chapter 3.

In this work we present an efficient floorplan-aware 3D NoC synthesis algorithm, based on simulated allocation (SAL), a stochastic method for traffic flow routing, and accurate power and delay models for NoC components. We demonstrate that this method finds greatly improved solutions compared to a baseline algorithm reflecting prior work. To evaluate the SAL method, we compare its performance with the widely-used simulated annealing (SA) method, and show that SAL is much faster than SA for this application while providing solutions of very similar quality. We then extend the approach from single-path routing to multipath routing scheme, and explore the tradeoff between power consumption and runtime for these two schemes. Finally, we study the impact of various factors on the network performance in 3D NoCs, including the TSV count and the number of 3D tiers. Our studies show that link power and delay can be significantly improved when moving from a 2D to a 3D implementation, but the improvement flattens out as the number of 3D tiers goes beyond a certain point.

2.1 Introduction

Three dimensional (3D) integrated circuits, in which multiple tiers are stacked above each other and vertically interconnected using through-silicon vias (TSVs), are emerging as a promising technology for SoCs [10–13]. As compared to 2D designs, 3D circuits permit reduced latencies for critical interconnect structures, resulting in higher system throughput, performance, and power, and allow other benefits such as heterogeneous integration. All of these flexibilities enable the design of new high-performance System-on-Chip (SoC) structures that were previously thought to have prohibitive overheads. In spite of well-known challenges such as thermal bottlenecks (to which several solutions have been proposed), the benefits of 3D integration are considerable. In the context of intrachip communication, 3D technologies have created significant opportunities and challenges in the design of low latency, low power and high bandwidth interconnection networks.

In 2D SoCs choked by interconnect limitations, networks-on-chip (NoCs), composed of routers and links, have been proposed as a scalable solution to the global communication challenges: compared to previous architectures for on-chip communication such as bus-based and point-to-point networks, NoCs have been shown to provide better predictability, lower power consumption and greater scalability [14, 15].

3D circuits enable the design of more complex and more highly interconnected systems: in this context, NoCs promise major benefits, but impose new constraints and limitations. Compared to wire interconnects, NoCs not only enable scalable and parallel communication within and across 3D tiers, but also reduce the number of TSVs for vertical interconnects. However, 3D NoC design introduces new issues, such as the technology constraints on the number of TSVs that can be supported, problems related to optimally determining tier assignments and the placement of routers in 3D circuits, and accurate power and delay modeling issues for 3D interconnects.

This work addresses the problem of designing application-specific 3D NoC architectures for custom SoC designs, in conjunction with floorplanning. Specifically, our work determines both the NoC topology and the floorplan of the NoC routers and cores. We propose a synthesis method to find the best topology for the application, under different optimization objectives such as power and network latency, and determine the paths for traffic flows. We use a 3D thermally-aware floorplanner to assign the cores to different 3D tiers, while optimizing chip temperature, and find an initial floorplan for the cores on each tier. Given the positions of cores, we use a stochastic flow allocation method, Simulated Allocation (SAL), to route the traffic flows and build the topology for the application, initially using a simple strategy for determining the approximate locations of the routers. When the best topology is found, a fast floorplanner is applied to further optimize the positions of the added routers. Accurate power and delay models for routers and links are integrated into our algorithm.

Our approach has three significant features that together make it uniquely different from competing approaches: first, we use improved traffic flow routing using SAL that accommodates a realistic objective function that has components that are nonlinear and/or unavailable in closed form; second, we interleave floorplanning with NoC synthesis, using specific measures that encourage convergence by discouraging blocks from moving from their locations in each iteration; and third, we use an accurate NoC delay model that incorporates the effects of queueing delays and network contention.

NoC synthesis can be based on either single-path or multipath routing: single-path routing can guarantee in-order delivery of packets and is much simpler to implement; multipath routing can exploit path diversity to evenly distribute the traffic across the network and to relieve traffic congestion, but the packets are sent in out-of-order fashion and re-ordering mechanism are needed at the re-convergent nodes [16]. We demonstrate that our SAL approach can work with either single-path or multipath routing scheme.

Our algorithm is extremely flexible and is applicable both to 2D and 3D layouts, but we demonstrate that the use of 3D designs results in significantly reduced NoC power and latency, when compared to optimal 2D implementations.

2.2 Contributions of This Work

There has been a great deal of prior work on NoCs alone and on 2D and 3D layout alone, but less on integrating the two. In the area of designing NoC architectures for 3D ICs, most of the literature has focussed on regular 3D NoC topologies such as meshes [17–21], which are appropriate for regular 3D designs [22, 23]. However, most modern SoC architectures consist of heterogenous cores such as CPU or DSP modules, video processors, and embedded memory blocks, and the traffic requirements among the cores can vary widely. Therefore, regular topologies such as meshes may have significant area and power overhead [24, 25], and tuning the topology for application-specific solutions can provide immense benefits.

The synthesis of an application-specific NoC topology includes finding the optimal number and size of routers, establishing the connectivity between the routers and with the cores, and finding deadlock-free routing paths for all the traffic flows. For 2D systems, the problem of designing application-specific NoC topologies has been explored by several researchers [16, 26–29]. Srinivasan *et al.* [27] present a three-phase NoC synthesis technique consisting of sequential steps that floorplan the cores, next perform core-to-router mapping, and then generate the network topology. In [16], Murali *et al.* present an NoC synthesis method that incorporates the floorplanning process to estimate link power consumption and detect timing violations. Several topologies, each with a different number of routers, are explored, starting from one where all the cores are connected to one router, to one where each core is connected to a separate router. The traffic flows are ordered so that larger flows are routed first.

In the 3D domain, Yan *et al.* [24] present an application-specific 3D NoC synthesis algorithm that is based on a rip-up-and-reroute procedure for routing flows, where the traffic flows are ordered in the order of increasing rate requirements so that smaller flows are routed first, followed by a router merging procedure. Murali *et al.* [25] propose a 3D NoC topology synthesis algorithm, which is an extension to their previous 2D work [16], described above. The 3D NoC synthesis problem has been shown to be NP-hard in [30].

Our work is motivated by the following observations:

- The final results of application-specific NoC topology synthesis depend on the order in which the traffic flows are routed. In some cases, routing larger flows first provides better results [16, 28], while in others, routing the smaller flows first may yield better results [24]. A strategy is required to reduce the dependency of the results on flow ordering.

- In all of the works mentioned previously, the average hop count is used to approximate the average packet latency in NoCs. This ignores the queueing delays in router ports and the contention among different packets for network resources such as router ports and physical links, and cannot reflect the impact of physical core-to-router or router-to-router distances on network latency. More accurate delay models that include the effects of queueing delay and network contention, and better delay metrics, should be applied for NoC performance analysis.

- The delays and power dissipation for physical links in NoCs are closely linked to the physical floorplan and topology of cores and routers. We show in Section 2.6 that interleaving floorplanning and NoC topology synthesis process leads to superior results.

We address these important problems in application-specific NoC topology synthesis. Our solution to overcoming the ordering problem is based on the use of a multicommodity flow network formulation for the NoC synthesis problem: the advantage of such an approach is that it takes a global view of the problem and eliminates the problem, described above, of finding the best order in which to route the traffic flows. The multicommodity flow problem is a well-known approach for solving such problems, but has seen little use in NoC design, with a few exceptions. In [31, 32], Hu *et al.* propose a scheme to optimize NoC power consumption through topology exploration and wire style optimization, subject to the average communication latency constraints, but do not handle layout synthesis issues, and assume simple linear objective functions.

Our work utilizes a stochastic SAL approach to efficiently solve the multicommodity flow problem under a nonlinear objective function that can be evaluated by an oracle, but is hard to express in closed form. The SAL framework has previously been used to solve multicommodity flow problems in computer network design. We also use an accurate delay model for routers in NoCs which consider the queueing delay and network contention. Finally, our algorithm performs the floorplanning of cores/routers and NoC topology synthesis in an integrated iterative loop, attempting to find the optimal solution for the problem of application-specific NoC design.

In the context of synthesizing application-specific 3D NoC architectures for custom SoC designs, this work makes the following contributions:

- We present an efficient floorplan-aware 3D NoC synthesis algorithm, based on simulated allocation, a stochastic method for traffic flow routing, and accurate power and delay models for NoC components. The effects of these strategies have been verified by the experiment results.

- We perform a comparative study between single-path and multipath routing schemes in the SAL framework. Simulation results show that tradeoff exists between single-path and multipath routing systems in terms of network power consumption and the efficiency to solve the multicommodity flow problems.

- We also compare our stochastic SAL approach with simulated annealing (SA). Our results show that SAL is much faster than SA to find approximately the same quality solutions.

- After that, we present the impact of TSV count on the network performance in 3D NoCs. Our results show that within certain extent, TSV count can be effectively reduced with mild penalty on the network performance.

- Finally, we investigate the impact of 3D integration on the NoC architecture design. Our studies show that link power and delay can be largely improved when moving to 3D implementation, at the cost of the TSV area and chip temperature. We also observe that the improvement on link delay and power flattens out as the number of 3D tiers goes beyond a certain point.

2.3 Problem Inputs, Objectives, and Constraints

The input to our application-specific 3D NoC synthesis problem is a directed graph, called the core graph, $G(V, E, \lambda)$. Each node $v_i \in V$ represents a core (either a processing element or a memory unit) and each directed edge $e_{v_i,v_j} \in E$ denotes a traffic flow from source v_i to destination v_j. The bandwidth of traffic flow from core v_i to v_j is given by $\lambda(e_{v_i,v_j})$ in MB/s. In addition, NoC architectural parameters such as the NoC operating frequency, f , and the data link width, W, are also assumed to be provided as inputs. The operating frequency is usually specified by the design and data link width is dictated by the IP interface standards.

Our 3D NoC synthesis framework permits a variety of objectives and constraints, including considerations that are particularly important in 3D, such as power dissipation, temperature, and the number of TSVs, and NoC-specific issues such as minimizing the average/maximum network latency, limitations on the maximum bandwidth, as well as general factors such as the design area. In addition, the solution must be free of deadlocks, which can occur during routing flows due to cyclic dependencies of resources such as buffers. We use the turn prohibition algorithm presented in [33] to ensure that our topology is deadlock-free. The specific optimization objectives in each step of our approach are described in Section 2.4.

The output of our 3D NoC synthesis solution is an optimized custom deadlock-free network topology with pre-determined paths on the network to route the traffic flows in the core graph and the floorplan of the cores and routers in the NoC such that the constraints are satisfied.

2.4 The Overall Design Flow

The design flow of our NoC synthesis algorithm is presented in Figure 2.1.

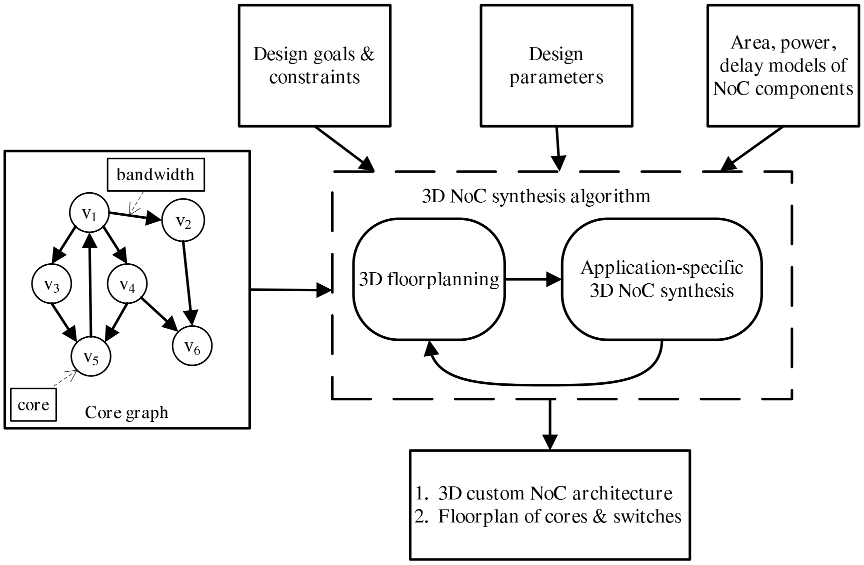

Design goals & constraints
Design parameters
Area, power, delay models of NoC components
bandwidth
v_1
v_2
v_3
v_4
v_5
v_6
core
Core graph
3D NoC synthesis algorithm
3D floorplanning
Application-specific 3D NoC synthesis
1. 3D custom NoC architecture
2. Floorplan of cores & switches

Figure 2.1: Application-specific 3D NoC synthesis flow.

Given a core graph, we first obtain an initial floorplan of the cores using a thermally-aware floorplanner. This precedes the 3D NoC synthesis step, and is important because the core locations significantly influence the NoC architecture. Associating concrete core positions with the NoC synthesis step better enables it to account for link delays and power dissipation.

Our 3D NoC synthesis algorithm is performed on a directed routing graph $G'(V',E')$: V' is the vertex set, which is the union of core set V in the input core graph $G(V,E,\lambda)$ and the set of added routers, V_s. We assume that the maximum number of routers that can be used in each 3D tier l equals to the number of cores in that tier, although it is easy to relax this restriction. The edge set E' is constructed as follows: we connect cores in a tier l only to the routers in the same tier l and adjacent tiers $l-1, l+1$ and the routers from all the 3D tiers form a complete graph. A custom NoC topology is a subgraph of the routing graph, G'.

The 3D NoC synthesis problem can be viewed as a *multicommodity flow* (MCF) problem. For a core graph $G(V,E,\lambda)$ and a corresponding routing graph $G'(V',E')$ (corresponding to a flow network), let $c(u,v)$ be the capacity of edge $(u,v) \in E'$. The capacity $c(u,v)$ equals to the product of the operating frequency f and data link width W. Each commodity $K_i = (s_i, t_i, d_i), i = 1, \cdots, k$ corresponds to the weight (traffic flow) along edge e_{s_i,t_i} in the core graph from source s_i to destination t_i, and $d_i = \lambda(e_{s_i,t_i})$ is the demand for commodity i. Therefore, there are $k = |E|$ commodities in the core graph. Let the flow of commodity i along edge (u,v) be $f_i(u,v)$. Then the MCF problem is to find the optimal assignment of flow which satisfies the constraints:

$$\begin{array}{rc}
\text{Capacity constraints:} & \sum_{i=1}^{k} f_i(u,v) \leq c(u,v) \\
\text{Flow conservation:} & \sum_{\omega \in V', u \neq s_i, t_i} f_i(u,\omega) = 0 \\
 & \text{where} \quad \forall v, u\, f_i(u,v) = -f_i(v,u) \\
\text{Demand satisfaction:} & \sum_{\omega \in V'} f_i(s_i,\omega) = \sum_{\omega \in V'} f_i(\omega,t_i) = d_i
\end{array}$$

Superficially, this idea seems similar to [32], where an MCF formulation is proposed. However, that work is directed to 2D NoC synthesis with a single objective of minimizing NoC power, modeled as a linear function of the flow variables $f_i(u,v)$. The corresponding Linear Programming (LP) problem is solved using an approximation algorithm. Our more general formulation integrates more objectives and more accurate modeling for NoC components. In fact, most components of our objective function are nonlinear or, as in case of network latency, unavailable in closed form, rendering an LP-based approach impossible.

We choose to apply an SAL-based flow allocation approach that is particularly suitable for (see Section 2.5.1 for details) solving the MCF problems where the objective function is in such a form. The SAL procedure yields the NoC topology and the paths for all the traffic flows in the core graph. In our work, we first present the SAL approach using single-path routing, and then show how to extend it to deal with the multipath routing problem in the experimental section.

After the 3D NoC synthesis step, the actual routers and links in the synthesized 3D NoC architecture are fed back to the floorplanner to update the floorplan of the cores and used routers, and the refined floorplan information is used to obtain more accurate power and delay estimates. The process continues iteratively: with the refined floorplan, a new SAL based 3D NoC synthesis procedure is invoked to find a better synthesis solution, and so on.

The specific optimization objectives used in various steps of our approach are as follows:

- For the initial floorplanning step, we optimize a linear combination of chip temperature and weighted inter-core distance (Section 2.5.4).

  $$\texttt{Objective cost} = w_1 * \texttt{temperature} + w_2 * \texttt{inter-core distance} \tag{2.1}$$

  where $w_1 = 1, w_2 = 5$ are default weights.

- For NoC topology construction, we optimize a linear combination of the network power, average network latency and TSV count, with constraints on link bandwidth.

  $$\texttt{Objective cost} = w_1 * \texttt{power} + w_2 * \texttt{latency} + w_3 * \texttt{TSV count} \tag{2.2}$$

  where $w_1 = 10, w_2 = 5$ and $w_3 = 3$ are default weights.

- For subsequent steps that floorplan the cores and routers, we optimize a linear combination of design area, link power, link delay and chip temperature.

$$\texttt{Objective cost} = w_1 * \texttt{area} + w_2 * \texttt{power} + w_3 * \texttt{delay} + w_4 * \texttt{temperature} \quad (2.3)$$

  where $w_1 = 10, w_2 = 5, w_3 = 3$ and $w_4 = 1$ are default weights.

In equations (2.1)-(2.3), we normalize the metrics such as power and latency using their initial numbers from a preliminary solution of the NoC synthesis. In a practical setting, the weights of these metrics in each cost function are user-specified, and can be chosen depending on the emphasis that the user wishes to place on each of these metrics.

2.5 Technical details

In this section, we present the major elements in our 3D NoC synthesis algorithm. We first introduce the SAL algorithm, the approach to synthesize the NoC topology, in Section 2.5.1. In Sections 2.5.2 and 2.5.3, we present the delay model and the method to estimate path cost used in our SAL algorithm. Finally we introduce the 3D floorplanner for the initial floorplanning step and subsequent floorplan refinement of cores and NoC routers.

2.5.1 Simulated Allocation Algorithm

Simulation Allocation (SAL) [8, 34] is a stochastic approach for finding near-optimal solutions for the multicommodity traffic flow problems in computer network design. It has been shown to be simpler, but often faster and more efficient, than other stochastic algorithms such as simulated annealing and evolutionary algorithms. We adopt the SAL framework from [34], but adapt it to solve the 3D NoC synthesis problem in our work. The details of the SAL algorithm used in our work are described in Algorithm 1.

In the core graph $G(V, E, \lambda)$, let

- P_i be the number of available paths for traffic demand $K_i = (s_i, t_i, d_i)$,

ALGORITHM 1: Simulated Allocation (SAL)

```
$n = 0$; $counter = 0$; $x = 0$; $F^{best} = +\infty$ ;
repeat
    if $random(0,1) < q(|x|)$ then
        allocation($x$);
    end
    disconnect($x$);
    if $|x| = H$ then
        $n = n + 1$;
        $counter = counter + 1$;
        if $F(x) < F^{best}$ then
            $F^{best} = F(x)$;
            $x^{best} = x$;
            $counter = 0$;
        end
    end
until $n = N$ or $counter = M$;
```

- x_{ip} be the amount of traffic flow realizing the traffic $K_i = (s_i, t_i, d_i)$ allocated to path p in routing graph G',
- $x = \{x_{ip} : i = 1, 2, \cdots, k, p = 1, 2, \cdots, P_i\}$ be the allocation state,
- $|x| = \sum_i \sum_p x_{ip}$ be the total allocated traffic flow, and
- $H = \sum_i d_i$ be the total amount of traffic flow.

Note that in this section, we use single-path routing to introduce how the SAL method works. In Section 2.6.3, we extend SAL to deal with multipath routing problems. For single-path routing, we assume that each commodity is non-bifurcated, and in the routing graph, at most k paths, one per commodity, will have nonzero flows. Therefore, even though the number of paths can be exponentially large, it is never necessary to enumerate P_i; storing the allocation state x does not impose a significant memory overhead.

The SAL algorithm may start with a given partial allocation state x_0 or with the zero state ($x_{ip} \equiv 0$). In each step, it chooses, with state-dependent probability $q(|x|)$, between *allocation*(x), i.e., adding the traffic flow for one non-allocated commodity to the current state x, and *disconnect*(x), i.e., removing the traffic flow for one allocated commodity from current state x. After a sequence of such moves, from time to time, the algorithm will reach a full allocation state, yielding a feasible solution for the considered problem. The procedure terminates when the number of visited full allocation states reaches a user-specified limit N or no better solution is found within M visited full allocation states.

Procedure *allocation*(x) selects one currently non-allocated commodity, $K_i = (s_i, t_i, d_i)$, at random and allocates it to one of the allowable paths that have enough residual capacity to support K_i in the routing graph. The path for allocating K_i is chosen to be the minimum cost path p with respect to the cost function for the NoC topology construction step. Then we add flow $x_{ip} = d_i$ to the current state x and reduce the capacities of the links on the selected path p in the routing graph by d_i. When routing commodity K_i, several new links and routers from the routing graph may be added to the NoC topology and the sizes of the routers on the path p may need to be adjusted accordingly.

Procedure *disconnect*(x) selects an allocated commodity $K_i = (s_i, t_i, d_i)$ at random and removes the corresponding flow x_{ip} from current state x. We then increase the capacities of the links on the path p by d_i. If some links/routers become unused in the resulting solution, such links/routers are also removed from the NoC topology. The sizes of the routers on the path p may need to be adjusted accordingly.

Function $q(\gamma)$, defined for $0 \leq \gamma \leq H$, has the properties:

$$\begin{cases} q(0) = 1 \\ q(H) = 0 \\ \frac{1}{2} < q(\gamma) \leq 1, \quad 0 < \gamma < H \end{cases}$$

According to [34], if

$$q(|x|) = q_0 > \frac{1}{2} \quad \texttt{for} \quad 0 < \gamma < H$$

then the expected average number of steps (allocations and disconnections) required to reach a full allocation state starting from state x is no greater than

$$(H - |x|)/(2q_0 - 1)$$

For instance, if $q_0 = \frac{2}{3}$ then a full allocation state will be reached from the zero allocation state in only $3H$ steps.

2.5.2 Analytical Router Delay Modeling for NoCs

Accurate delay models for routers are required as an input to our 3D NoC synthesis problem, since we need the models to 1) estimate the router delay when routing a traffic flow in the *allocation(x)* step in Section 2.5.1, and 2) evaluate the final 3D NoC synthesis solutions. In our work, we utilize the analytical delay model presented in [35], which includes the effects of queueing delay and network contention. The model considers first-come-first-serve input buffered routers and targets wormhole flow control under deterministic routing algorithms.

Let S be the packet size and H_i the service time for a header flit passing through router i. The service time of a packet passing through router i, excluding the queueing delay, is

$$T_i = H_i + \frac{S - W}{f \cdot W} \tag{2.4}$$

where W is the data link width and f is the operating frequency. For router i, let

- p be the total number of ports.
- λ_{ij} be the traffic arrival rate at port j.
- N_j be the average number of packets in the buffers of input port j, and $N = [N_1, N_2, \ldots, N_p]^T$.
- c_{jk} be the probability that packets of input ports j and k compete for the same output port, and C_j be the row vector $C_j = [c_{j1}, c_{j2}, \ldots, c_{jp}]$.
- R be the residual service time seen by the incoming packets, defined as follows: if another packet n is being served when packet m arrives, then R is the remaining time before packet n leaves the router.

Then we can write the *equilibrium condition* for the router as:

$$(I - T\Lambda C)N = \Lambda \bar{R} \tag{2.5}$$

where $\Lambda = diag\{\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{ip}\}$, $C = [C_1, C_2, \ldots, C_p]^T$, $\bar{R} = ([R, R, \ldots, R]_{1\times p})^T$.

The router model described by Equation (2.5) provides a closed form expression for the average number of packets at each input port of the router i, given the traffic arrival rate (Λ), the packet contention probabilities (C), router design specifications (H_i, W) and packet size S.

We further use this router model to compute the average packet latency from source core s to destination core d (used in Equation (2.2)) as:

$$L_{sd} = \sum_{i \in \prod_{sd}} (H_i + \tau_i) \; + \; D_{sd} \; + \; \frac{S - W}{f \cdot W} \tag{2.6}$$

where

- $\prod_{sd}$ is the set of routers along the path of the packets sent from source s to the destination d,
- τ_i is the average waiting time of the incoming packets at router i, which can be estimated as $\tau_i = N_j/\lambda_{ij}$ by Little's theorem [36],
- D_{sd} is the total link delay from s to d.

For further details, the reader is referred to [35].

2.5.3 Router Location Estimation and Path Cost Estimation

When routing a flow from source s to destination d in the *allocation(x)* step (refer to Section 2.5.1), our objective is to find a minimum cost path in the routing graph. While the initial solution considers the physical locations of only the cores, as flow allocation proceeds, new routers will be included in the NoC topology and their physical positions must be estimated to compute the link power and delay.

We estimate the router locations in the following way: for a newly added router i, the router is initially placed at the centroid of the source and destination nodes of router i in the routing graph. Given these initial estimates of the positions of the newly added routers, we apply Dijkstra's shortest path algorithm on the routing graph to find the minimum cost path for the traffic flow, which is required by *allocation*(x). Here the path cost is the cost for NoC topology synthesis, as shown in Equation (2.2). When the 3D NoC synthesis step is complete, we feed the actual routers and links in the synthesized architecture to the floorplanner to update the router locations, for more accurate power and delay estimation. Since the floorplanner is stochastic, it is possible for the new floorplan to be vastly different from the one that was used to generate the NoC topology, negating the assumptions used to build the topology. To avoid this, we add a penalty to the objective function of the floorplanner to ensure that the blocks do not move far away from their initial locations, and optimize the precise locations of the routers, which were initially placed in (possibly illegal) centroid locations.

2.5.4 3D Floorplanning

As described in Section 2.4, an initial step of thermally-aware floorplanning is applied to assign the cores into 3D tiers under thermal considerations, and to optimize the positions of the cores so that highly communicating cores are placed close to each other. In our implementation, we use the 3D thermally-aware floorplanner tool in [37] based on $B^\star$-tree floorplan model. The floorplanner uses a built-in thermal analysis technique based on the HS3D [37] tool. Of course, any other similar tools can also be integrated into our program.

For each edge e_{v_i,v_j} which connects two cores, v_i and v_j, the edge weight of e_{v_i,v_j} is set to be the product of edge bandwidth $\lambda(e_{v_i,v_j})$ and the distance d_{ij} between v_i and v_j. Our cost function is a weighted sum of the chip temperature and the sum of these edge weights. Therefore, we use the floorplanner to find a good initial floorplan of cores that favors our next step of 3D NoC synthesis.

During initial floorplanning, we only consider the communicating cores, since no routers have been introduced at this time. Once a full allocation of traffic flows is found, the topology of the NoC is determined, including the routers that are used to route traffic. We then invoke the floorplanner to find a refined floorplan of cores and NoC routers, under an objective function that is a linear combination of design area, link power, link delay and chip temperature.

2.6 Experimental Results

2.6.1 Experimental Setup

We have implemented *3D-SAL-FP*, our SAL-based 3D NoC synthesis algorithm with floorplan feedback, in C++. All experiments were conducted on an Intel Pentium 4 CPU 3.20GHz machine with 2G memory running Linux.

The design parameters are set as: 900MHz clock frequency, 512-bit packets, 4-flit buffers and 32-bit flits. We use Orion [38, 39] to estimate the power dissipation of the routers. The link power and delay are modeled based on the equations from Pavlidis *et al.* [18]. Considering that in 3D circuits the delay and power of TSV (interdie interconnect) is much smaller (at least one order in magnitude) than that of the intradie interconnect wires [24, 40], we ignore the delay and power of TSVs in this work. The delay of routers are estimated using the model described in Section 2.5.2. All routers and links are evaluated under a 45nm technology.

Several parameters affect the efficiency and performance of the SAL algorithm (Section 2.5.1). In choosing the $q(\gamma)$ function for SAL, we found that for $0 < \gamma < 1$, a constant function $q(\gamma) = q_0 = 0.9$, can produce good solutions. The user-specified iteration limit N is empirically set to be three times of k, the number of commodities in the core graph, and M is set to be 50 or 100, depending on the size of MCF problem. We find that the best solutions are often obtained within k visited full allocation states for all the benchmarks.

2.6.2 Impact of each strategy applied in our algorithm 3D-SAL-FP

Our algorithm *3D-SAL-FP* (based on single-path routing) improves upon the previous algorithms in [24, 25] by: 1) using a more sophisticated traffic flow routing algorithm (SAL), 2) adding a feedback loop of floorplanning and NoC synthesis to refine the NoC architecture, 3) using a more accurate router delay model including the effects of queueing delay and network contention. To show the separate impact of these techniques on the NoC design, we have implemented three other 3D NoC synthesis algorithms.

The four algorithms that we will compare in our results are:

- *Baseline 1*, based on the work by Murali *et al.* [25], has two stages: 3D NoC synthesis and floorplanning of the synthesized NoC architecture. At the 3D NoC synthesis stage, a simple delay model (average hop count) is used to approximate the average network latency and the traffic flows are routed in fixed order (in the order of decreasing flow rate). In the next stage, we find the floorplan of cores and used routers in the NoC architecture.

- *Baseline 2* differs from *Baseline1* in that it applies an improved traffic flow routing strategy (SAL) in the 3D NoC synthesis stage.

- *Baseline 3* improves upon *Baseline2* by feeding back the results of floorplanning stage to refine the NoC synthesis. The process continues iteratively: after the 3D NoC synthesis step, the actual routers and links in the synthesized solution is fed back to the floorplanner to refine the floorplan of the cores and used routers; with the refined floorplan, a new NoC synthesis procedure is invoked to find a better synthesis solution, and so on.

- *3D-SAL-FP* is our proposed approach, and differs from *Baseline3* in that it use the accurate router delay model (described in Section 2.5.2) to incorporate the queueing delay and network contention issues.

We then applied these four algorithms to design 3D application specific NoC topologies. We compared these algorithms on both a set of existing published benchmarks and several large synthetic 3D benchmarks. Since large standard benchmarks are not available, we use the method proposed in [24] to build large synthetic 3D benchmarks, which can be viewed as the "many-core" version of the small published ones. This method is based on the NoC-centric bandwidth version of Rent's rule proposed by Greenfield *et al.* [41]. For the small published benchmarks, two 3D tiers are used, where each tier contains one layer of devices and multiple layers of interconnect. For all of the large synthetic benchmarks, four 3D tiers are used.

The corresponding results are shown in Tables 2.1 and 2.2. For each algorithm, we report the following: the network power (in mW, including router power and link power), the average network latency (in ns, evaluated by the accurate delay model), the number of TSVs and the maximum chip temperature (in $^\circ C$). Considering that SAL is a stochastic approach, we run each algorithm for 10 times and present data in the

Table 2.1: Comparison of three algorithms on several small published benchmarks

Ben	Cores	Flows	*Baseline1*						*Baseline2*						*Baseline3*						*3D-SAL-FP*					
			Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}
			Router	Link	Total				Router	Link	Total				Router	Link	Total				Router	Link	Total			
PIP	8	8	54	5	59	3.8	8	66.4	44	4	48	3.7	6	60.6	39	4	43	3.6	7	58.2	38	4	42	3.2	6	55.1
MWD	12	13	94	8	102	4.1	10	72.8	74	7	81	4.0	9	66.5	65	6	71	3.8	12	62.5	65	6	71	3.5	9	62.3
VOPD	12	15	99	11	110	7.3	14	67.8	82	10	92	7.2	7	64.5	73	10	83	6.9	7	59.4	72	9	81	5.1	9	50.9
MEPG4	12	26	165	15	180	10.3	14	70.8	108	15	123	10.1	13	64.7	88	12	100	9.0	14	58.2	90	13	103	6.3	14	59.6
IMP	27	96	612	90	702	9.4	42	78.8	413	99	512	8.0	44	65.2	335	87	422	7.8	42	55.7	346	79	425	6.4	40	56.9
					1	1		1			0.77	0.95		0.90			0.66	0.91		0.82			0.66	0.74		0.80

Table 2.2: Comparison of three algorithms on large synthetic benchmarks

Ben	Cores	Flows	*Baseline1*						*Baseline2*						*Baseline3*						*3D-SAL-FP*					
			Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}
			Router	Link	Total				Router	Link	Total				Router	Link	Total				Router	Link	Total			
B1	56	196	1033	291	1324	16.3	119	157.8	956	302	1258	16.0	132	145.4	808	209	1017	15.0	139	128.3	785	214	999	6.7	132	133.2
B2	80	96	783	128	911	7.9	117	133.5	561	118	689	7.9	116	119.6	490	99	589	7.6	124	107.1	494	96	590	4.6	126	107.5
B3	69	136	866	210	1076	13.1	122	150.6	494	243	737	12.0	95	134.4	509	165	674	11.5	105	118.2	504	141	645	9.4	116	118.0
B4	114	396	3128	827	3955	15.9	196	166.4	2230	888	3118	15.5	214	151.6	1826	643	2469	13.9	192	128.6	1721	632	2353	7.3	208	137.0
B5	124	266	1827	848	2675	13.9	254	135.9	1517	686	2203	11.8	264	125.2	1352	432	1784	11.4	256	104.4	1338	468	1806	9.1	241	102.7
					1	1		1			0.79	0.94		0.91			0.66	0.89		0.79			0.65	0.56		0.80

tables showing the best results among all the runs. The same strategy is applied to the experiments in the subsequent sections.

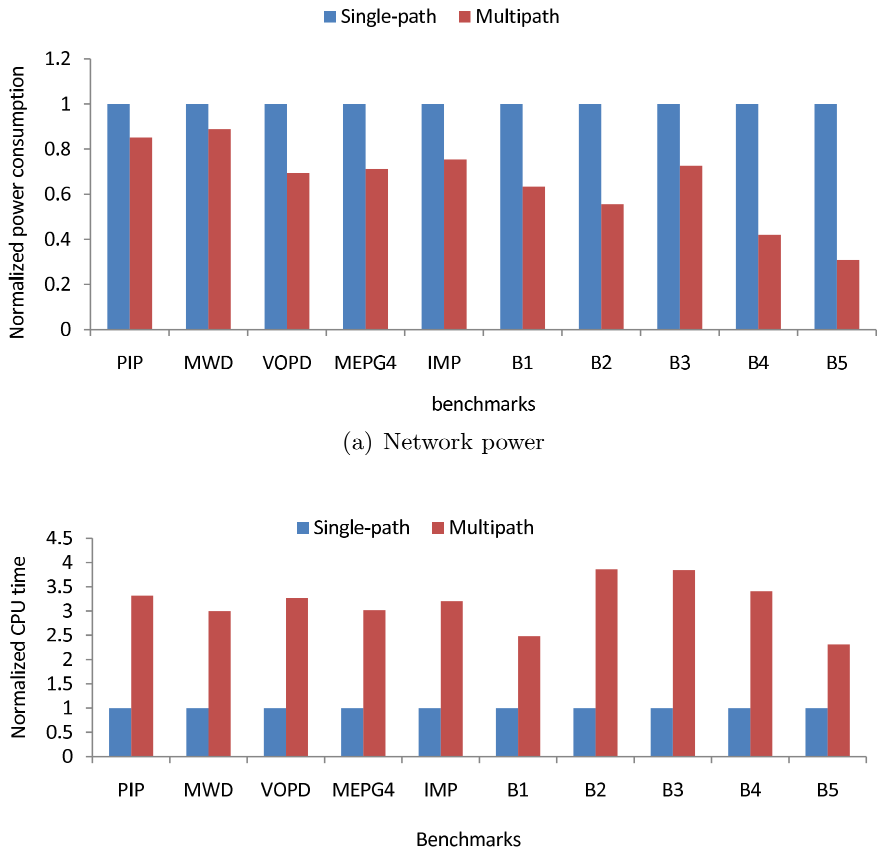
We can observe that using the improved traffic flow routing algorithm, the *Baseline2* algorithm outperforms *Baseline1*, achieving 23% power saving for the published benchmarks, 10% reduction in chip temperature and better network performance. The corresponding numbers for synthetic benchmarks is 21% in power saving and 9% in chip temperature reduction. Furthermore, *Baseline3* uses the feedback from the floorplanning step to improve upon *Baseline2*, and shows 34% reduction in the power dissipation for both published and synthetic benchmarks, about 20% reduction in chip temperature and 10% reduction in average network latency. Finally, with more accurate delay model, *3D-SAL-FP* improves upon *Baseline3*, with 26% reduction in average network latency for published benchmarks and 44% for the synthetic benchmarks. Since the objective function for these algorithms is a linear combination of several metrics, the use of different sets of weighting factors can result in different Pareto-optimal solutions. For a fair comparison, we have used identical weighting factors for all four algorithms discussed here. In the solutions shown here, *3D-SAL-FP* performs significantly better than *Baseline3* in reducing the delay, and is slightly better on average (and sometimes worse on specific examples) in terms of power and temperature. By altering the weights, other tradeoff points may be identified.

2.6.3 3D-SAL-FP Based on Multipath Routing

In Section 2.6.2, our *3D-SAL-FP* algorithm is based on single-path routing, which means that each commodity (traffic flow in the given core graph) is non-bifurcated and we choose one single path in the routing graph for one commodity. In this section, we extend *3D-SAL-FP* to work with multipath routing where each commodity can be split into several subflows and each subflow can be routed independently in the routing graph.

Let L be the capacity of each subflow, then a commodity with traffic demand d_i can be split into $\lceil d_i/L \rceil$ subflows. Here we use the capacity L to control the granularity of the subflow, so that the size of the MCF problem in multipath routing can be controlled. In our experiments, L is set individually for each benchmark because the values of the traffic demand d_i varies greatly from benchmark to benchmark. After splitting the commodities in the core graph, we treat each of the resulting subflows as a single routing unit and select one minimum cost path to route it. Since the subflows constituting the same commodity can be routed on different paths on the routing graph, we refer to the new routing problem supporting subflows as *multipath routing*. In fact, the costs for the routers and links in the routing graph are state-dependent, when the subflows are routed one after another, it is highly possible for the subflows of one commodity to be routed on different paths.

Figure 2.2 presents the comparison results of single-path and multipath routings.



(a) Network power

(b) CPU time

Figure 2.2: Comparisons of single-path and multipath routing schemes.

The results are normalized to single-path case. Considering that multipath routing can reduce the peak link bandwidth needs and therefore lower network operating frequency [16], we evaluate the network power consumption using the optimized frequency number corresponding to the peak link bandwidth in the NoC synthesis solution: Given the peak link bandwidth, we can obtain the optimized NoC frequency as optimized frequency = peak link bandwidth/link width.

From Figure 2.2 we can see that on average multipath routing can obtain 35% power savings compared to single-path routing. The overhead in run time for multipath routing is more than 3X for most of the benchmarks. This is because 1) it takes longer time for SAL to find a full allocation solution in multipath routing, 2) SAL needs to explore an expanded solution space for the multipath case. However, this overhead is related to the increased search space, and will affect any other algorithm (e.g., SA) that solves this problem formulation.

2.6.4 Comparison of SAL and Simulated Annealing

In this section, we compare the performance of our single-path based SAL algorithm with another widely-used stochastic approach, Simulated Annealing (SA), by replacing SAL with SA in the *3D-SAL-FP* implementation. We implement two kinds of SA moves in our work:

1. Consider that in Algorithm 1, SAL approach applies two basic moves *allocation*(x) and *disconnect*(x). In order to perform a fair comparison, we integrate these two basic moves into the SA engine: in SA, given one full allocation state x, a move to a neighbor full allocation state *neighbor(x)* is implemented as a series of single-flow moves, *allocation*(x) and *disconnect*(x), as introduced in Section 2.5.1. We refer to this SA implementation as *Single-flow SA*.

2. Given one full allocation state x, a move to a neighbor full allocation state *neighbor(x)*, can be obtained in one of the two ways:

    - Disable one of the used routers, and reroute all the traffic flows passing through that router.
    - Disable one of the used links, and reroute all the traffic flows passing through that link.

For this SA implementation, several flows may be rerouted in one single move, so we refer to it as *Multi-flow SA*.

The performance and runtime of SA algorithm is affected by several parameters, such as the initial and end temperature T_i and T_e, the inner loop number N_{inner} at each temperature and temperature reduction parameter τ. We investigate the impact of runtime on SA's performance with one randomly selected benchmark *IMP*. Figure 2.3 shows the simulation results when *Single-flow SA* is applied.

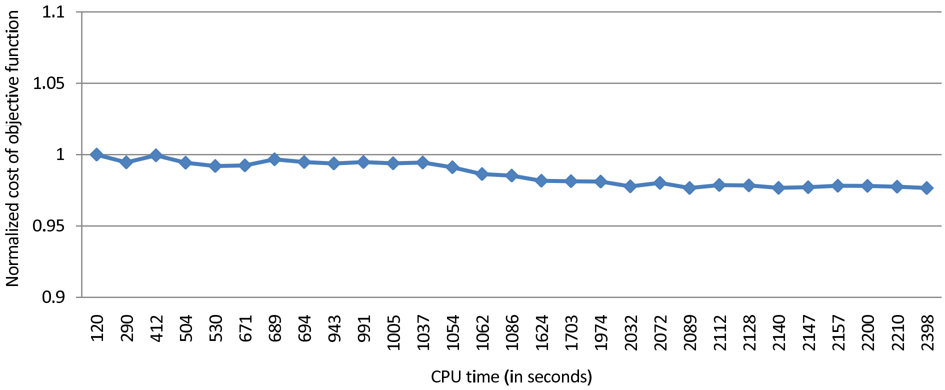


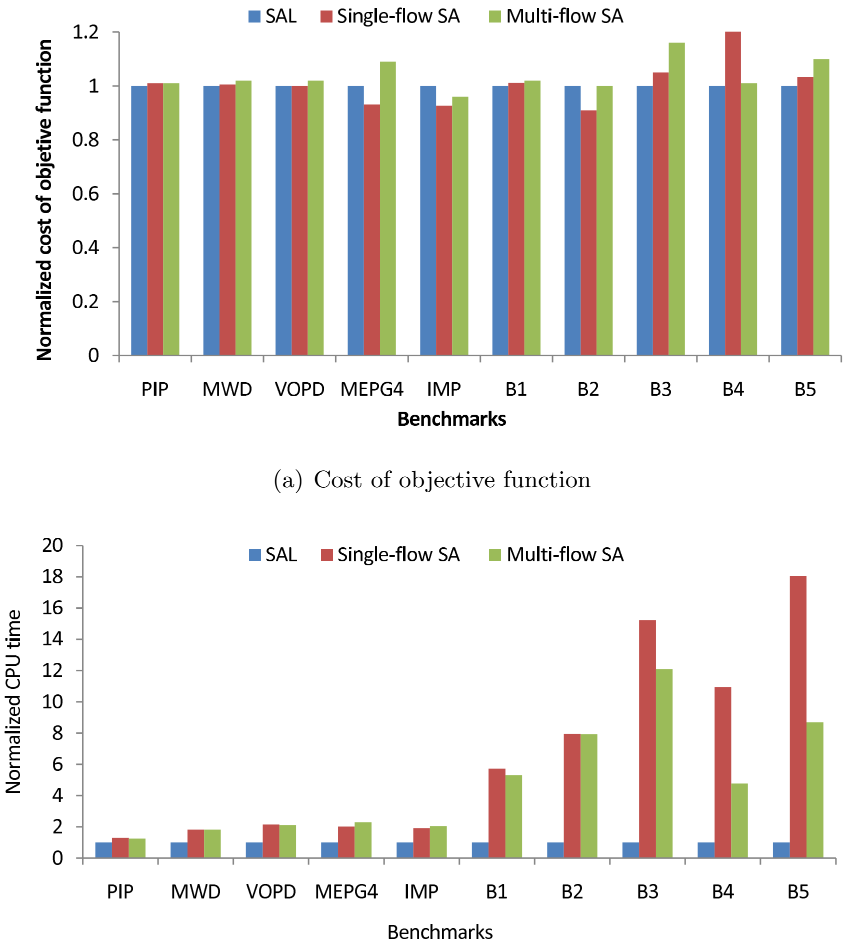
Figure 2.3: The impact of runtime on the performance of SA for benchmark IMP.

We use the following cost function to evaluate the final 3D NoC solutions:

$$cost = w_1 * \texttt{chip area} + w_2 * \texttt{network power} + w_3 * \texttt{network latency} + w_4 * \texttt{TSV count} \tag{2.7}$$

the default weights are $w_1 = 1, w_2 = 10, w_3 = 5 \, \texttt{and} \, w_4 = 3$. We normalize all the costs to the baseline case with runtime of 120 seconds. We tune the parameters for an appropriate runtime/quality tradeoff from SA. For example, for benchmark *IMP*, Figure 2.3 shows that as we increase the runtime of SA from 120 seconds to 2398 seconds, the improvement to the objective function shows diminishing returns: the improvement is very small but the increase in the runtime is about 20X. We factor this into our experiments, and the runtime of SA for the benchmarks ranges from several minutes to several hours.

Figure 2.4 presents the results of SAL and two SA implementations on both the published and synthetic benchmarks.



(a) Cost of objective function

(b) Runtime

Figure 2.4: Comparisons with SA.

We use the cost function shown in Equation (2.7), where all results are normalized to SAL case. Figure 2.4 shows that in terms of the quality of the solutions, SAL performs approximately as well as SA, but that the execution times are much smaller than those of SA. For example, for the large benchmark B5 with 124 cores and 266 flows, the cost reported by SAL is about 3% less than that of *Single-flow SA*, while the speedup is about 18X. Compared to *Multi-flow SA*, *Single-flow SA* has longer execution time for most of the benchmarks because it needs more moves to find a neighbor full allocation state, but it can find slightly better solutions in most cases.

2.6.5 Exploration of TSV Count

Next, we explore the tradeoffs associated with using more or fewer TSVs in the design. In 3D circuits, more TSVs imply more vertical interconnects, which mean that the latency can be reduced in the resulting NoC topology. However, the corresponding overhead includes increased design area and excessive utilization of a valuable vertical resource (note that TSVs are also required for routing supply nets, clock nets, thermal vias, etc.). In this section, we explore the tradeoff between TSV count and network latency. Single-path based *3D-SAL-FP* algorithm is applied for this experiment.

Figure 2.5 shows the tradeoff curve when we gradually increase the weight of TSV w_4 (Equation 2.2) from 1 to 18. As we can see from this figure, the number of TSVs can be largely optimized when we increase w_4 from 1 to 6, and the increase of the network latency is less than 2%. After that point, the TSV count gradually levels off and the network latency increases much faster. The minor nonmonotonicities in this figure can be attributed to the nature of the stochastic approach.

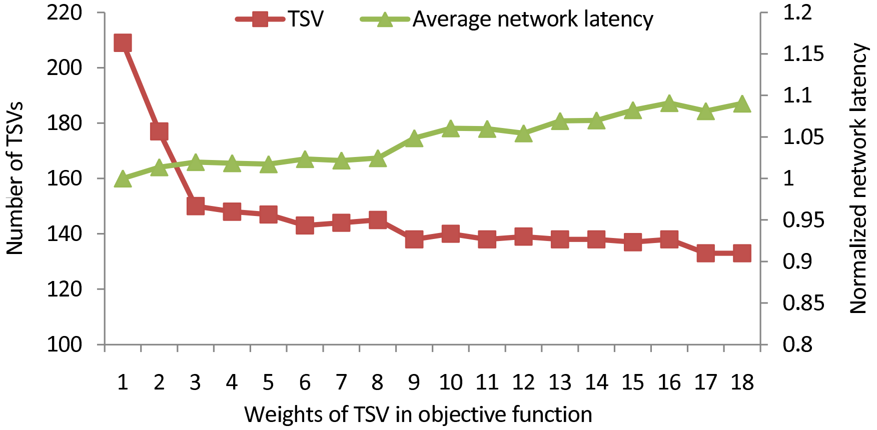


Figure 2.5: The tradeoff between number of TSV and average network latency for benchmark B1.

2.6.6 Delay and Power Reduction Potential in 3D NoCs

In this section, we further investigate the impact of 3D integration on the NoC architecture design. The benchmark B3, with 69 cores and 136 flows, was selected and our *3D-SAL-FP* algorithm was applied to synthesize this benchmark with different numbers of 3D tiers, from 1 to 4. The 1-tier case is the design that uses conventional 2D technology. The results are shown in Table 2.3. For each case, we list the following results: the design footprint, the network power, the maximum path length, the maximum total link delay, the maximum network latency, the average network latency, the total number of TSVs, the maximum chip temperature and the CPU time.

Table 2.3: Comparison of the impact of different numbers of 3D tiers on NoC architecture design for benchmark B3

#tier	Footprint	Network Power			Max Path	Max Link	Max Network	Avg Network	# of	T_{max}	Time
		Router	Link	Total	Length	Delay	Latency	Latency	TSVs		
	(mm^2)	(mW)	(mW)	(mW)	(mm)	(ns)	(ns)	(ns)		($^\circ C$)	(s)
1	216.8	510.5	288.4	798.9	22.1	6.45	14.40	12.42	0	43.8	85.8
2	110.3	505.8	189.2	695.0	17.0	4.95	12.28	9.56	86	63.7	83.9
3	72.0	510.7	164.8	675.5	11.9	3.50	11.51	9.49	94	96.2	87.3
4	56.1	504.8	141.0	645.8	9.2	2.68	11.32	9.44	116	118.0	87.4

Our results show the clear tradeoff when implementing NoC architecture using 3D circuits: as the number of 3D tiers increases, the footprint size continues to decrease, together with the maximum length of the path to route the packets. The reduced path length further brings down the maximum link delay and the total link power at the cost of increased number of TSVs and higher chip temperature. In addition, we can observe that although 3D circuits have the potential to reduce the link delay and power, the improvement flats out as the number of 3D tiers goes beyond a certain point. For example, as shown in Table 2.3, the network latency does not decrease much as we go from three tiers to four.

2.7 Conclusion

In this chapter, we have proposed an efficient algorithm, *3D-SAL-FP*, to synthesize application-specific 3D NoC architectures. Our algorithm utilizes a stochastic approach called simulated allocation (SAL) to reduce the dependency of NoC design results on flow ordering. We also use accurate delay model for routers in NoCs which consider the queueing delay and network contention. Finally, our algorithm performs the floorplanning of cores/routers and NoC topology synthesis in an integrated iterative loop, attempting to find the optimal solution for the problem of application-specific NoC design.

Experimental results on a set of benchmarks show that our algorithm can produce greatly improved solutions compared to the baseline algorithm with fixed-order flow routing, simple delay model and without feedback from floorplanning step, reflecting prior work. In comparison with SA, we show that SAL can find approximately the same quality solutions, but with better computational efficiency.

We have also investigated several degrees of freedom in this space. First, our comparative study between single-path and multipath routing schemes in the SAL framework shows that multipath routing can achieve large power savings with slightly larger computation times. Second, when we study the impact of TSV count on the network performance in 3D NoCs, we find that there is a “sweet spot” where the TSV count is effectively controlled without much penalty on the network performance. Third, we investigate the benefits that 3D circuits can bring to the NoC architecture design, and show that link power and delay can be largely improved when moving to 3D implementation, at the cost of TSV and chip temperature.

Chapter 3

NoC Frequency Scaling with Flexible-Pipeline Routers

In this chapter, we consider NoC optimization problem for CMP applications. As presented in Chapter 1, CMPs show intermittent or "bursty" traffic patterns, leading us to conclude that methods that adaptively scale the voltage and frequency (VFS) of the network (providing it just as much performance as necessary and saving power while doing so) can be used very effectively to potentially reduce NoC energy consumption.

Although VFS for NoC can potentially reduce energy consumption, but the associated increase in latency and degradation in throughput limits its deployment. In this work, we propose a hardware technique, called flexible-pipeline router, that reconfigures pipeline stages upon VFS, so that latency through such router remains constant. With minimal hardware overhead, the deployment of such routers allows us to reduce network frequency and save network energy, without significant performance degradation. Furthermore, we demonstrate the use of simple performance metrics to determine the optimal operation frequency, considering the energy/performance impact on all aspects of the system - the cores, the caches and the interconnection network.

3.1 Introduction

Advances in semiconductor technology have led to continuous increases in device density and larger system sizes. Concomitant with exponentially reducing device dimensions, designers face new challenges in maximizing computation while remaining with a stringent power envelope. Over the last decade, CMPs have emerged as a potential solution to address some of these problems by integrating multiple smaller and more energy efficient cores in order to replace a single, larger core. These cores must communicate through an efficient on-chip interconnection network (NoC), and NoC design is vital to both performance and power.

If incorrectly designed and/or poorly utilized, NoCs can become a major performance bottleneck and a significant source of power consumption for CMP systems [7, 42, 43]. As CMP-based systems become the main powerhouse for computation, they must serve diverse computing needs; and thus the on-die NoCs must be designed for a variety of traffic patterns. The integration of heterogeneous cores [43, 44] onto a single die further aggravates this situation, since cores with different computation capability have different performance goals. By identifying the performance requirements of each core, it might be possible to reduce the energy consumption of the NoC, while achieving the same overall performance.

State-of-the-art NoC designs often use packet-switched routers to support high bandwidth traffic. Under this model, it often takes multiple hops for messages to reach their destinations, and the energy/delay associated with packets traversing through routers is the dominating factor. There have been several proposals for reducing the performance penalty, such as router bypassing [45–47] and enhancing router pipeline design [48–50]. There also exists a large body of work on reducing router energy consumption, which corresponds to a large portion of NoC energy [7, 42].

A critical design parameter that directly affects both performance and power of NoC is the network frequency. Techniques such as VFS [51–54] have been widely investigated to allow the network to operate at a lower frequency to reduce energy consumption, when possible. Of which [51–53] specifically focus on link latency or power. Mishra *et al.* [54] use VFS on the NoC routers to reduce the network power consumption. However, reducing NoC frequency increases latency and reduces network throughput, which in turn, degrades overall system performance. An unfortunate side-effect of performance degradation is that the processors must be kept active for a longer duration, causing them to consume more leakage power: this factor has become increasingly dominant in nanometer-scale technologies. As a result, in prior work, frequency is only moderately scaled, by up to 20%, with the network and the cores operating asynchronously at different frequencies. To fully realize the potential of VFS, it is desirable to be able to scale down the network frequency significantly (our work examines changes of 2×-4×) up to the point where the performance penalty remains small. Moreover, since our frequency scaling uses integer multiples of the clock frequency, we can continue to operate within a fully synchronous paradigm.

Our performance analysis indicates that different applications and cores are sensitive to different NoC performance metrics. For some applications, a reduction in NoC throughput has relatively little impact on performance, but increasing NoC latency can cause significant performance degradation. For these types of workloads, we propose to reconfigure the router pipeline when scaling down the network frequency. With this technique, the impact on the average time to traverse the NoC for a single message is minimal, while the peak throughput of the NoC degrades. Thus, for workloads that have a low or moderate throughput requirements, but are sensitive to the NoC latencies, this technique allows us to reduce NoC energy consumption, without significant performance degradation. We refer to the proposed routers as *flexible-pipeline* routers, since the router pipeline stages are adaptively configured based on the workload.

To demonstrate the effectiveness of the proposed technique, we build a flexible-pipeline router based on a classic 4-stage baseline pipeline. In general, when NoC utilization is low, NoC frequency is scaled down. Meanwhile, the router pipeline is reconfigured by combining some pipeline stages, while using the same latch boundaries. We use time-borrowing techniques to ensure that this delay-unbalanced reconfigured pipeline runs at the optimal clock period. We evaluate the proposed router in a chip multiprocessor connected by a mesh NoC. Our results demonstrate that for a large class of applications that are only sensitive to network latency, NoC frequency can be dramatically scaled down without significant performance degradation. This leads to improvement in NoC energy efficiency.

Any solution that uses pipelined routers for energy savings must satisfy two requirements: (1) The hardware overhead must be low: The overhead for our flexible-pipeline router is minimal, and it only requires a set of multiplexers to bypass registers and alter clock skews. Our analysis in Section 3.2.2 shows that the cost of the additional hardware is less than 3% compared to the classic 4-stage baseline router. (2) The savings must be demonstrable at the system level: We evaluate the impact of flexible pipeline routers in the context of an entire multicore system, including the cores, the caches and the interconnection network, and present the system-wide performance and energy-delay^2 (ED^2) product. We find that static energy consumption corresponds to a significant portion of energy consumption. When lowering router frequency, system performance can degrade, and the weight of static energy increases in total energy consumption, leading to the conclusion that VFS is unable to consistently improve system ED^2.

Related work by Hirata *et al.* [55] presented the idea of variable-pipeline (VP) routers, which aims to achieve similar functionality. However, when the VP router is evaluated using the above two criteria: (1) its hardware overhead is substantial (13%), and (2) it only simulated VP routers with network components under a simplified zero-load latency model and did not provide system-level conclusions.

In the context of improving NoC energy efficiency, this work makes the following contributions:

- We propose a flexible-pipeline router, where the number of pipeline stages can be dynamically reconfigured. We reduce the number of pipeline stages when scaling down the NoC frequency. The proposed router enables us to scale down the network frequency without increasing router latency.

- We deploy the flexible-pipeline routers in a homogeneous CMP system connected by a mesh network, and use realistic workload to evaluate their performance impact when the network frequency is scaled down to reduce network energy consumption.

- We find that VFS may improve or degrade system ED^2. Therefore, we propose to use simple performance metrics to determine whether VFS should be applied.

The rest of this chapter is organized as follows: In Section 3.2, we discuss our pipeline reconfiguration approach for the flexible pipeline. The experimental platform and workload used in this study are described in Section 3.3. We present our experimental results in Section 3.4 followed by concluding remarks in Section 3.5.

3.2 Flexible router pipeline design

We use a classic four-stage-pipelined virtual-channel router as an example to show how our strategy works. However, our approach can definitely work on other enhanced router designs so long as they have multiple pipeline stages. In this section, we first provide an architectural overview of the classic router. We then introduce our procedure for optimizing the pipeline design to maximize the benefit obtained while speed-tuning the NoC.

3.2.1 Baseline Router Architecture

Figure 3.1 illustrates the microarchitecture of a classic fixed four-stage-pipelined virtual-channel (VC) router with p input/output ports, as used in Garnet [56]. The major components that constitute a router are the input buffers, the route computation logic, the VC allocator, the switch allocator, and the crossbar switch.

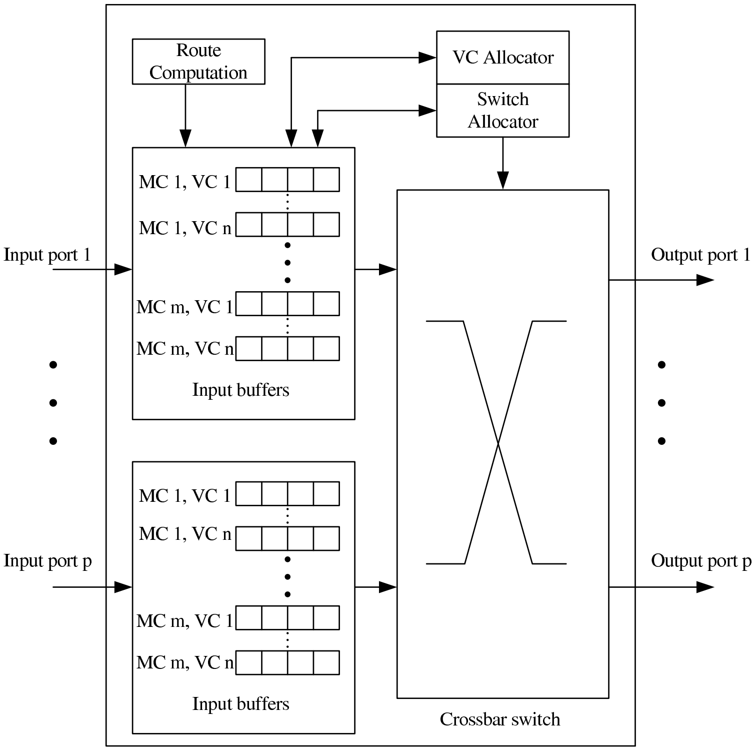


Figure 3.1: Classic four-stage virtual-channel router

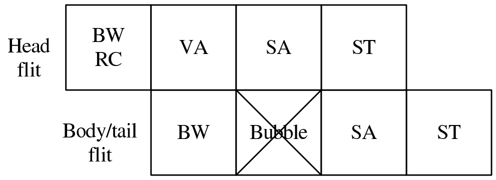


Figure 3.2: Router pipeline

Figure 3.2 shows the corresponding fixed four-stage router pipeline. Since on-chip designs must adhere to tight budgets and low router footprints, flit-level buffering and credit-based VC flow control are applied for every router. The router supports multiple message classes (MCs), and VCs from all MCs are multiplexed across the input port. Every VC has its own private flit buffer and its size can be specified at runtime. The routing is table-based and deterministic.

When a head flit arrives at an input port, it is first decoded and buffered in the buffer write (BW) pipeline stage, according to its input VC ID. In the same cycle, a request is sent to the route computation unit (RC) simultaneously, and the output port for this packet is calculated based on the destination information presented in each head flit. The header then arbitrates for a free VC of its output port in the VC allocation (VA) stage. Upon successful allocation of an output VC, it proceeds to the switch allocation (SA) stage where it arbitrates for the switch input and output ports. On winning the switch, the flit moves to the switch traversal (ST) stage, where it traverses the crossbar and is placed on the output link connected to the next node. The body and tail flits follow a similar pipeline except that they do not go through RC and VA stages, instead inheriting the VC allocated by the head flit. The tail flit deallocates the VC reserved by the packet when it leaves the router.

3.2.2 Flexible-pipeline Router

Although the fine-granularity router pipeline design introduced in Section 3.2.1 can run at high frequency, and therefore support high throughput, it may degrade the system performance significantly when the network is slowed down to save power consumption. This is because such a change causes the network latency to increase, which has undesirable effects as discussed in Section 3.1. Therefore, we propose a flexible-pipeline reconfiguration approach to adapt to a change in the network speed, based on accurate delay models for the components in the baseline router introduced in Section 3.2.1.

Delay Models of Router Components

We model the delay of each router component shown in Figure 3.1 by the technology-independent parametric equations presented in [48].

For each component i, we have two delay estimates: *latency* (t_i) and *overhead* (h_i). Defined precisely, the *latency* is the time from when inputs are presented to the component to when the outputs needed by the next component are stable. The *overhead* refers to the setup delay expended by additional circuitry required before the next set of inputs can be presented to the component. As we introduce our timing model, let

- τ be the delay of an inverter with identical input capacitance (at 65nm technology, SPICE simulations show that the value of τ is 7.8ps@1.2V),
- p be the number of input/output ports,
- c be the number of message classes,
- v be the number of VCs per message class, and
- w be the flit size in bits.

The technology-independent parametric delay equations for each router component are listed in Table 3.1 [48].

Table 3.1: Parameterized delay equations (in τ) for baseline router

Component	Parametric delay equation
BW + RC(BR)	$t_{BR} = 100$, $h_{BR} = 0$
VA	$t_{VA} = 16\frac{1}{2}\log_4 pv + 16\frac{1}{2}\log_4 v + 20\frac{5}{6}$, $h_{VA} = 9$
SA	$t_{SA} = 11\frac{1}{2}\log_4 p + 23\log_4 cv + 20\frac{5}{6}$, $h_{SA} = 9$
ST	$t_{ST} = 9\log_8(w\lfloor\frac{p}{2}\rfloor) + 6\lceil\log_2 p\rceil + 6$, $h_{ST} = 0$

Flexible Router Pipeline Reconfiguration

Based on the delay models shown in Table 3.1, we can further determine the optimal number of pipeline stages, for various router sizes and various operating frequencies and supply voltages for the NoC.

We first introduce how the stage delay, T, is calculated for a pipeline stage that includes a few sequential components. Let a and b be the first and last components in the pipeline stage. Given the t_i and h_i of each component i on the critical path, we have

$$T = \sum_{i=a}^{b} t_i + h_b \tag{3.1}$$

For example, if we combine the stages SA and ST in the baseline router into a single stage, then the delay for combined stage would be $T = t_{SA} + t_{ST} + h_{ST} = 11\frac{1}{2}\log_4 p + 23\log_4 cv + 9\log_8(w\lfloor\frac{p}{2}\rfloor) + 6\lceil\log_2 p\rceil + 26\frac{5}{6}$ in units of τ.

Table 3.2 lists the t_i and h_i numbers of each component for the cases of a 5-port router and a 6-port router.

Table 3.2: Delay values (in units of τ) of each router component

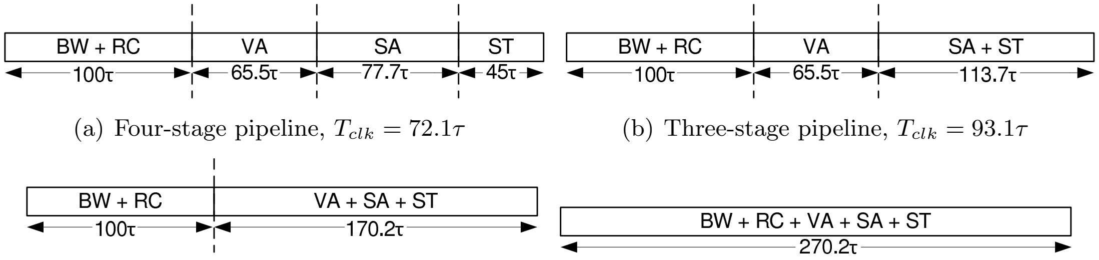
Router	BW+RC(BR)		VA		SA		ST	
	t_{BR}	h_{BR}	t_{VA}	h_{VA}	t_{SA}	h_{SA}	t_{ST}	h_{XB}
5-port	100	0	56.5	9	68.7	9	45.0	0
6-port	100	0	58.7	9	70.2	9	46.8	0

From this data we can clearly see that the pipeline stages are imbalanced, and conventional clocking would set the period to correspond to the pipeline stage with the longest delay. Instead, in our work, we apply the time-borrowing techniques to boost the pipeline frequency using clock skew optimization [57], where slower stages in the pipeline borrow time from faster stages, such that the linear router pipeline can operate at the average cycle time of all the pipeline stages. Let T_i be the delay for pipeline stage i and let n be the total number of stages. Then the clock time for the n-stage router after time-borrowing is

$$T_{clk} = \frac{\sum_{i=1}^{n} T_i}{n} \tag{3.2}$$

Furthermore, this four-stage linear router can be reconfigured as either a three-stage, or a two-stage, or even a one-stage pipeline. For example, a three-stage pipeline can be obtained by combined any of the two successive components such as 1) BR and VA, or 2) VA and SA, or 3) SA and ST. Of these, we choose the one with the minimum optimized clock period T that maximizes the router frequency that the pipeline can support: note that since the h_i values for various stages are different, the optimal frequency varies with our choice. The best choices are found to be: 1) for the three-stage pipeline, we combine SA and ST into one single stage and 2) for the two-stage pipeline, we combine VA, SA and ST into one single stage.

Figure 3.3 shows our final pipeline reconfiguration results for a 5-port router, and Table 3.3 summarize the results for two different router sizes after applying time-borrowing techniques. For each pipeline design case, we list the optimal clock time T_{clk} in units of τ, and the maximum frequency F that the pipeline can support in the GHz range. To achieve energy saving, we also apply voltage scaling as we scale down the frequency and the following V_{dd} values are used: 1.2V for four-stage pipeline, 1.1V for three-stage pipeline, 1.0V for two-stage pipeline and 0.8V for one-stage pipeline. Considering that the circuit delay is a function of the supply voltage V_{dd}, we scale τ accordingly based on Alpha-power law [58] for the above voltage settings.



(a) Four-stage pipeline, $T_{clk} = 72.1\tau$

(b) Three-stage pipeline, $T_{clk} = 93.1\tau$

(c) Two-stage pipeline, $T_{clk} = 135.1\tau$

(d) One-stage pipeline, $T_{clk} = 270.2\tau$

Figure 3.3: Optimal pipeline reconfiguration for a 5-port router, time borrowing technique is applied to boost the pipeline frequency.

Table 3.3: Optimal clock periods/frequencies for various pipeline configurations

Router	4-stage		3-stage		2-stage		1-stage	
	T_{clk}	Max. F	T_{clk}	Max. F	T_{clk}	Max. F	T_{clk}	Max. F
5-port	72.05	1.78	93.07	1.32	135.10	0.87	270.2	0.39
6-port	73.43	1.75	94.90	1.30	137.85	0.85	275.7	0.38

Given the maximum frequency numbers for each pipeline design, it is straightforward to select the optimal pipeline design at a given network speed. The basic principle behind this idea is that at the same network speed, a shorter pipeline with fewer stages leads to lower router latency, and therefore, better network performance. For example, for the 5-port router, if its frequency is set to be 1.5GHz, then only the four-stage pipeline with a peak frequency of 1.78GHz can be fast enough. However, if we reduce the router frequency to 1.0GHz, then both four-stage pipeline and three-stage pipeline can meet the frequency demand, but we will choose three-stage pipeline to maximize the router performance. Table 3.4 presents the results of optimal pipeline stage number $N = 1, 2, 3, 4$ for different routers with different router to processor clock ratio S (explained in Section 3.3.1).

Table 3.4: The optimal number, N, of pipeline stages with different processor to router clock ratio S; the processor frequency is 1.5 GHz.

Router	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$
5-port	4	2	2	1	1
6-port	4	2	2	1	1

Architecture Support for Flexible Pipeline Reconfiguration

The hardware support required for out flexible-pipeline router is minimal, and only needs a set of multiplexers that bypass registers and alter clock skews. In Figure 3.4 we only show the multiplexers used to bypass the registers after each pipeline stage. Given a particular pipeline configuration, we set the stage selection signals to combine the successive stages correspondingly. The delay of the 2:1 multiplexers used in the flexible-pipeline router is just several τs, therefore their impact on the pipeline clock period is negligible for all the pipeline configurations. For the area cost, after analyzing the gate count of each component in the flexible-pipeline router, we find that the router area is dominated by the input buffers and VC allocators, and the multiplexers only account for less than 2% hardware overhead in terms of the router area.

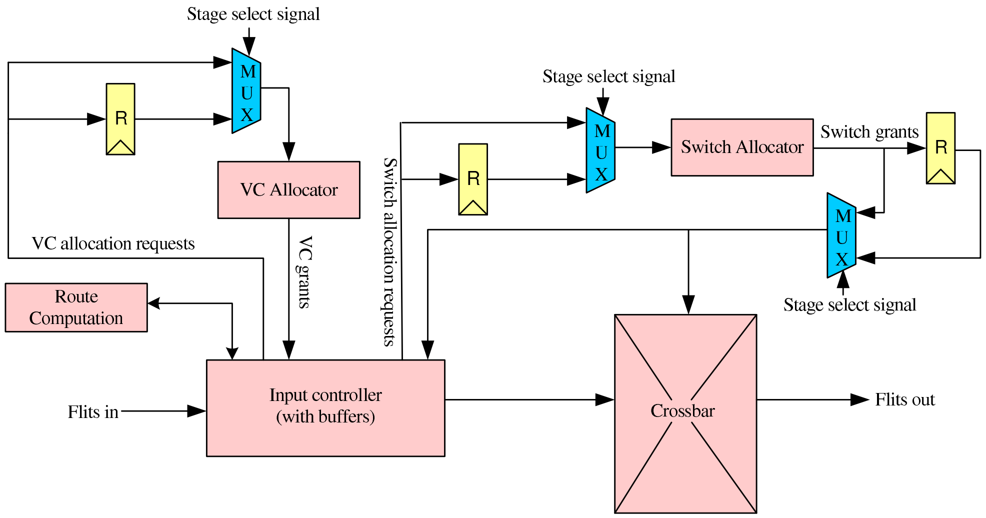


Figure 3.4: Router architecture for flexible pipeline reconfiguration

3.3 Experimental platform

3.3.1 CMP System Simulator

We use Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) [59] as our simulation engine. GEMS is a Simics-based [60] full system simulator using timing-first simulation approach. This infrastrcture provides detailed performance and power models for the processor pipeline, the memory hierchy, as well as the NoC. The NoC simulator provides the flexibility for customizing the interconnection by providing support for various network topologies, including user-specified ones. The NoC power model is provided by Orion 2.0 [39], which reports the total switch and link power. The NoC power breaks down into three segments: dynamic, static (leakage) and clock power.

In this work, we simulate an 8-core CMP system with 65nm technology nodes, where the cores/L2 cache banks and memory controllers are connected using a mesh network. Architecture parameters can be found in Table 3.5. The interconnection network uses a deterministic shortest path routing algorithm. For the processor power, we employ Wattch [61], an architectural level power modeling tool. We have updated the technology-specific parameters in Wattch based on the ORION 2.0 technology file. Simulation results show that the power of floating function units (FUs) dominates the

Table 3.5: Baseline Simulation Configuration

Processor Core	1.5GHz, one-way in-order 3 integer FUs, 6 floating FUs
Private L1 Cache	Split private I/D caches, each 2KB, 2-way set associative, 64B block size, 1-cycle access latency
Shared L2 Cache	4M banked, shared distributed, 512KB (per core) 4-way set associative, 64B block size, 8-cycle access latency
Memory	4GB DRAM, 200 cycle access latency, four memory controllers (one in each corner node)
Router	5 Input/output ports, 4-stage Pipeline, 5 message classes, 2 VCs per class, 64-bit flits, 4-flit buffer depth, 1 flit per control packet, 9 flits per data packet, wormhole routing

total FU power. Therefore, we use power gating and clock gating techniques to disable the inactive floating FUs in the CMP system.

In our baseline, the processors and NoC operate at the same frequency. Upon frequency scaling, we introduce a slow down parameter S, which is the router-to-processor clock ratio. For example, $S = 2$ implies that one network clock cycle is equivalent to two processor cycles. In our experiment, we experiment with three integer values, $S =$ 1, 2, and 4. We found that, the link delays are small enough that for any of the S values used here, a flit can traverse a link within one network cycle.

3.3.2 Workloads

We choose applications from the SPEC OMP2001 [62], NU-Mine [63] and PARSEC [64] benchmark suites as our workload input. In total we evaluate 6 correctly-compiled benchmarks written in C/C++. An overview of these applications are shown in Table 3.6.

In accordance with common practice in the architecture community when working with such benchmarks, in order to reduce simulation time, for each benchmark we first fast-forward to the beginning of the region of interest (the parallel section representative

Table 3.6: Benchmark Descriptions

ammp	Computational Chemistry
art	Neural network simulation; adaptive resonance theory
blackscholes	Computational finance
equake	Finite element simulation; earthquake modeling
fkmeans	Fuzzy-logic based data partitioning
kmeans	Mean based data partitioning

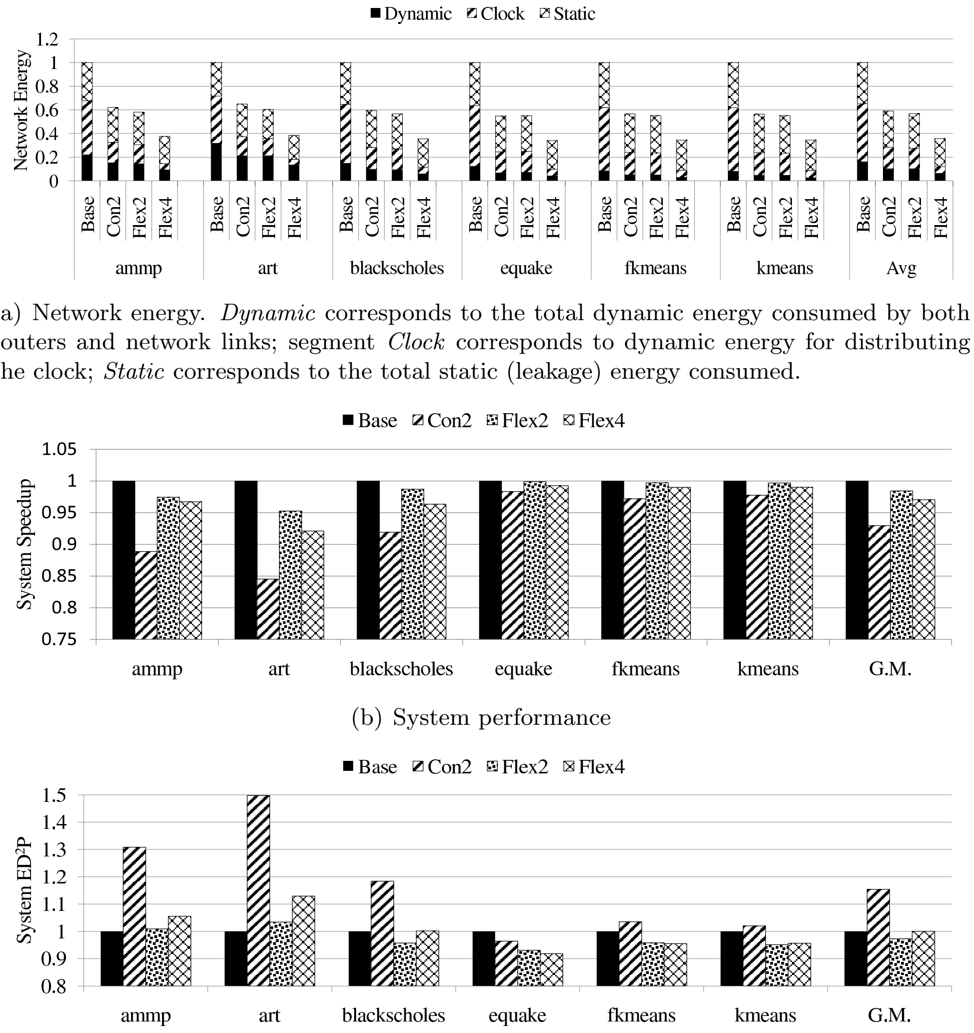
of the whole application) in Simics without loading Ruby; after that, we load Ruby and simulate one billion instructions.

3.4 Experimental Results

In this section, we evaluate a CMP, as described in Section 3.3.1, with the proposed flexible-pipeline routers, and compare its performance and energy consumption with a system that uses fixed-pipeline routers. Figure 3.5 shows the performance and energy results, with routers scaled with different slow down factors ($S = 2, 4$). For the reasons specified in Section 3.1, we limit the choice of S to integer values. The results are normalized to the case where the NoC is unscaled i.e., $S = 1$.

In general, reducing the network frequency can reduce energy consumption (Figure 3.5(a)), but it can also increase network latency and degrade network throughput. Such impacts directly translate into system performance degradation, as shown in Figure 3.5(b). The first set of bars in Figure 3.5(b) corresponds to the unscaled NoC (bars *Base*), the next two sets of bars correspond to CMP systems with fixed-pipeline routers (bars *Con2*) and flexible-pipeline routers (bars *Flex2*) when the network frequency is scaled by 50% and voltage is scaled from 1.2V to 1.0V.

For fixed-pipeline routers (bars *Con2*), on average, energy consumption is reduced by 41%, but system performance is degraded by 7%. The reduction in energy consumption is mainly due to reduction in clock energy and dynamic energy. In particularly, clock energy reduction due to frequency scaling is significant. This is because clock energy is proportional to the product of V_{dd}^2 and number of clock transitions. For example, when the network voltage is reduced from 1.2V to 1.0V and the network frequency is slowed



(a) Network energy. *Dynamic* corresponds to the total dynamic energy consumed by both routers and network links; segment *Clock* corresponds to dynamic energy for distributing the clock; *Static* corresponds to the total static (leakage) energy consumed.

(b) System performance

(c) ED^2 of the system

Figure 3.5: Comparison of fixed-pipeline and flexible-pipeline routers. *Base* corresponds to no scaling and using fixed-pipeline routers. *Con2* corresponds to network frequency scaled down by a factor of two, and using fixed-pipeline routers. *Flex2* and *Flex4* corresponds to network using flexible-pipeline router and frequency scaled down by a factor of two and four, respectively. All results are normalized to *Base*.

down by a factor of 2, it corresponds to a 65% reduction of clock energy. Dynamic energy is proportional to V_{dd}^2 and the amount of work performed by the NoC. Since the data path that each packet traverses is the same and the number of packets is similar before and after scaling; the amount of work performed by the NoC is not significantly changed due to scaling. As a result, dynamic energy of the network decreases quadratically as network voltage goes down. Static energy of the network reduces linearly with network voltage but increases linearly with total execution time. In our experiment, changes in network static energy are small compared to clock and dynamic energy.

When the flexible-pipeline router is deployed, with the same voltage/frequency scaling, the average system performance degradation is only 1.6%, and the average network energy consumption is reduced by 43% (bars *Flex2*). This corresponds to a 2% additional reduction from the fixed pipeline case. This small reduction in energy is a side effect of performance improvement: by completing the application faster, we are able to reduce static energy. Overall, NoCs with flexible-pipeline routers are more energy efficient.

The performance impact of network voltage and frequency scaling varies with the workload. Table 3.7 lists the cache miss rates for all the workloads.

Table 3.7: Cache Miss Rates For Evaluated Workloads

Workload	L1 data cache (misses/K instructions)	L2 cache (misses/K instructions)
ammp	13.7	4.4
art	40.8	18.1
blackscholes	8.1	0.9
equake	2.8	2.6
fkmeans	1.9	1.7
kmeans	2.4	1.9

Some applications, such as *ammp* and *art* suffer from a large number of L1 cache misses, and thus are very sensitive to L2 cache response time through the network. For these applications, network frequency scaling can degrade performance in two ways. On the one hand, frequency scaling leads to increases in network latency and L2 cache response times, which in turn increases the number of cycles per instruction (CPI). On the other hand, network frequency scaling decreases the throughput capability of the network, and thus causes more contentions. Although flexible-pipeline routers are able to effectively avoid increasing network latency in the absence of network contention, they see some level of throughput degradation due to contention. Thus, applications with high throughput requirements are likely to suffer performance degradation with frequency scaling. For other applications, when frequency is scaled down, deploying flexible-pipeline routers allows us to reduce energy consumption without suffering appreciable performance degradation.

If the network contention is low, flexible-pipeline routers make it possible to scale down network frequency without increasing the latency. However, this technique cannot prevent degradation in the network throughput, and thus frequency scaling can still lead to performance degradation. Bars 1, 3 and 4 of Figures 3.5(a) and 3.5(b) show the performance and network energy consumption as a result of voltage and frequency scaling when the network is scaled by factors of two and four, respectively, again using the no scaling case as a reference. It is clear that scaling reduces both the performance and the energy. When network is scaled by a factor of four, all workloads suffer noticeable performance degradation. However, the energy savings in going from a scaling factor of two to four shows diminishing returns as compared to the case where we go from scaling factor one to two. This is primarily related to the fact that applications take longer to complete, and the NoC hardware is activated for a longer duration, during which static (leakage) energy is expended. This increase in static energy offsets the gains made in reducing the dynamic network energy.

Up to this point, we have only considered the energy consumption of the network. However, whether VFS is beneficial must be considered in the context of the entire system, including the cores, the caches and the interconnection network. Figure 3.5(c) shows the ED^2 of the entire system. In comparison to flexible-pipeline routers, system with fixed-pipeline routers has higher ED^2 due to significant system performance degradation. We also find that for applications with a high cache miss rate, such as *ammp* and *art*, frequency scaling degrades ED^2 even with flexible-pipeline routers. This is because network throughput degrades with frequency scaling, even when flexible-pipeline router is used; and the above applications have high throughput requirement. As a result, these applications suffer from performance and ED^2 degradation upon frequency scaling. Furthermore, extended execution time also leads to more static energy consumption, which in turn further degrades ED^2.

The decision of scaling down network frequency should be specific to each application. In our case, the four out of six benchmarks benefit from frequency scaling. For some applications, such as *blackscholes*, aggressive frequency scaling adversely degrades ED^2. This is because for *blackscholes*, when the scaling factor is increased from 2 to 4, the decrease of system energy (0.4%) cannot offset the increase of CPI (2.4%). As a result, in some cases it is undesirable to aggressively scale down network frequency. It is worth pointing out that cache miss rate is only one metric for determining optimal router frequency. Other performance metrics, such as router utilization, can also serve as good indicators.

3.5 Conclusion

This chapter presents flexible-pipeline routers that are capable of re-balancing the pipeline stages upon voltage and frequency scaling, while operating the cores at the original frequency. The hardware complexity for supporting pipeline rebalancing is minimal. We compare 8-core mesh-based CMP systems with fixed- and flexible-pipeline routers running realistic workloads, respectively. When frequency is scaled down, energy reduction is dramatic for both systems, while the performance degradation can be low to medium. We compute the system ED^2 and show that for some benchmarks this value is noticeably improved with frequency scaling, while for others, it degrades. The improvement is application-specific and we see a close relationship with the cache miss count. However, for systems with fixed-pipeline routers, both throughput and latency degrades; while for systems with flexible-pipeline routers, latency remains unchanged as throughput degrades. Thus, as long as the network throughput is low, frequency-scaling on systems with flexible-pipeline routers are able to achieve energy saving with minimal performance degradation; and thus are more energy-efficient.

The proposed routers are able to improve energy-efficiency of the system by exploiting the fact that certain workloads are latency-sensitive, but are not throughput-intensive. The proposed technique can work in tandem with (and are largely orthogonal to) other techniques that are intended to reduce router power, such as router bypassing [45–47], router pipeline bypassing [50], speculative switch arbitration [48], as well as various congestion management strategies [54]. For routers that support bypassing and speculative arbitration, it is still possible to re-balance the pipeline stages, but we must take care to only re-balance the non-bypassed pipeline stages. Existing congestion control mechanisms can work in NoC with flexible-pipeline routers, although network bottlenecks may shift as a result of frequency scaling.

Chapter 4

Power Grid Optimization in 3D Circuits Using MIM and CMOS Decoupling Capacitors

In the previous two chapters, we present work on the on-chip communication network for 3D and multicore processors. In this and following two chapters, we explore another aspect – the design and optimization of on-chip power delivery network.

In three-dimensional (3D) chips, the amount of supply current per package pin is significantly more than in two-dimensional (2D) designs [9]. Therefore, the power supply noise problem, already a major issue in 2D, is even more severe in 3D. CMOS decoupling capacitors (decaps) have been used effectively for controlling power grid noise in the past, but with technology scaling, they have grown increasingly leaky. As an alternative, metal-insulator-metal (MIM) decaps, with high capacitance densities and low leakage current densities, have been proposed. In this chapter, we explore the tradeoffs between using MIM decaps and traditional CMOS decaps, and propose a congestion-aware 3D power supply network optimization algorithm to optimize this tradeoff. The algorithm applies a sequence-of-linear-programs based method to find the optimum tradeoff between MIM and CMOS decaps. Experimental results show that power grid noise can be more effectively optimized after the introduction of MIM decaps, with lower leakage power and little increase in the routing congestion, as compared to a solution using CMOS decaps only, and motivate the stronger need for these decaps in 3D technology, as compared to 2D designs.

4.1 Introduction

Three dimensional (3D) circuit technologies, with multiple tiers of active devices stacked above each other, are a key approach to increased levels of integration and performance in the future. However, there are two significant limitations that 3D technologies must overcome before achieving their full potential, related to on-chip thermal issues and reliable power delivery. Both issues can be illustrated through a simple back-of-the-envelope calculation. A k-tier 3D chip that stacks k similar chips could use k times as much current as a single 2D chip of the same footprint. However, the packaging technology is not appreciably different: with a similar heat sink, the on-chip temperature on such a 3D chip can be up to k times higher than the 2D chip, and with a similar number of pins in the package, the current per pin is k times higher than the 2D case.

The above analysis operates under very coarse assumptions (for example, a smart 3D designer may not stack k layers with identical power levels), and a more nuanced approach is necessary for a more accurate analysis – but the eventual conclusions that thermal and power delivery issues are important in 3D – are inescapable. While much research has been conducted on thermal management strategies such as thermal via insertion, and the spatial distribution of power sources, the power delivery problem has attracted limited attention to date.

The power delivery problem can be summarized as follows. The parasitics in the power network, together with temporal variations in the current drawn by a circuit, result in a time-varying voltage drop/surge at nodes in the power grid. These variations can adversely impact the performance and the reliability of a circuit. Such shifts become more acute with technology scaling: on the one hand, noise margins become more stringent with reducing Vdd levels, and on the other hand, with increased switching speeds and larger currents, IR, LdI/dt, and electromigration effects become more prominent. In 3D circuits, robust power supply network design is more challenging, and significant resources have to be invested in building a bulletproof power grid for the 3D chip.

Several techniques are available to increase the reliability of power grids and control power grid noise, such as wire widening, grid topology optimization, and decap insertion. Of these techniques, decaps are arguably the most powerful method for reducing transient noise, and are therefore addressed in this paper. Decaps serve as local current reservoirs, and can be used to satisfy sudden surges in current demand by the functional blocks/cells, while keeping supply voltage levels relatively stable.

Conventional technologies for implementing decaps are based on SiO_2-based structures that are widely used in robust power delivery network design. In the recent past, the CMOS decap allocation and optimization problem has been investigated by numerous researchers for 2D [65–67] and 3D technologies [67–69].

Unlike the 2D case, new considerations come into play while optimizing a 3D power grid using CMOS decaps. Since CMOS decaps are usually fabricated using white space on the device layer, they must compete for area with through-silicon vias, or with the landing pads of 3D vias, for the limited white space. This leads to a new resource contention problem. One way to resolve this contention problem is to increase the chip size in order to make room for CMOS decaps. However, one of the advantages of 3D circuits over 2D implementations is their reduced chip footprint: increasing the chip size may counteract this benefit. Leakage power is an important issue in 3D circuit design. The CMOS decaps added to the 3D circuit will consume extra leakage power, and make things worse. While new high-k dielectrics have been proposed, they will provide temporary relief to the gate leakage problem.

In this work, we address all of these issues. One of the novel features of our work is that it optimizes the power supply network using both conventional CMOS decaps and MIM decap technology, illustrated in Figure 4.1, which has high capacitance density and low leakage current density [2, 70, 71]. However, since MIM decaps are built between layers of metal interconnects, they present routing blockages to nets that attempt to cross them, and therein lies the tradeoff. The properties of MIM decaps makes them attractive for both 2D and 3D chips, but we pay particular attention to the 3D decap problem in this paper because (i) the power integrity problem is particularly critical in 3D, and requires novel approaches that leverage advances in materials, and (ii) the added complexity of handling routing blockages in a very constrained environment makes the 3D problem especially challenging.

We formulate the decap budgeting problem as a Linear Programming (LP) problem,

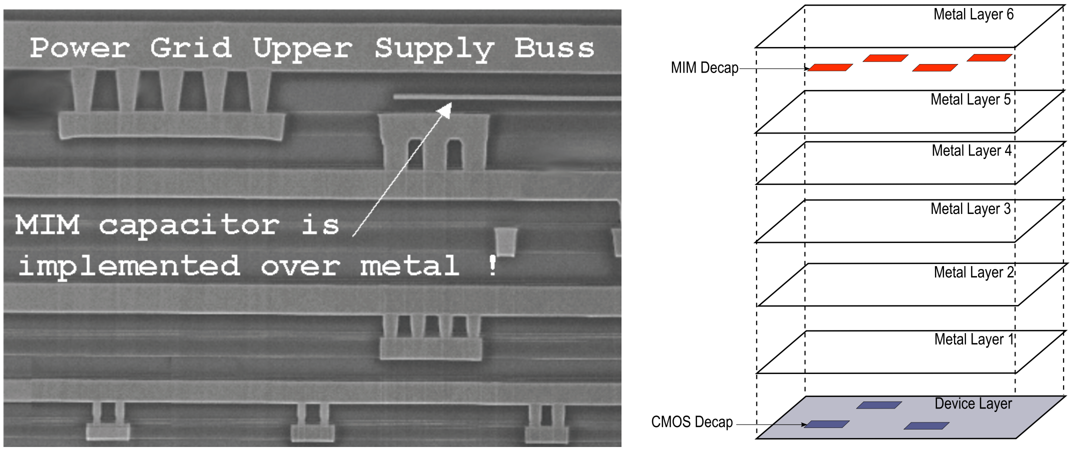


Figure 4.1: (a) Schematic of a MIM decap [2]. (b) MIM and CMOS decaps in one 2D tier with 6 metal layers.

and propose an efficient congestion-aware algorithm to optimize the power supply noise, while trying to find a balance between the routing congestion deterioration and leakage power increase.

4.2 Problem Formulation

We tile the layout using an uniform grid G' that is coarser than the original power grid, G, so that each tile of G' contains less than 20 power nodes in G. Our algorithm proceeds iteratively, adding a small amount of decap to the circuit in each iteration. An observation node is dynamically chosen from G for each tile in G', and all newly added decaps in this tile are connected to this observation node in each iteration. This helps in reducing the number of possible decap insertion spots, thus controlling the size of the problem that we solve.

4.2.1 Objective function

A key metric for the objective function is the noise violation area for the circuit. Given the transient voltage waveform, $v_i(t)$, at each node i of the power grid, the *violation* *area*, S_i, at the node is given by [65]:

$$S_i \;=\; \sum_j \int_{t_{s,j}}^{t_{e,j}} \max\{V_{limit} - v_i(t), 0\} dt \tag{4.1}$$

where, V_{limit} is the voltage threshold, usually set to be 90% of V_{dd}, and $[t_{s,j},\, t_{e,j}]$ is the j^{th} interval during which the constraint is violated. The noise violation area, S, is the sum of S_i over all nodes. The goal of our optimization is to reduce the violation area to zero at all nodes, with optimal resource usage.

We denote the newly added CMOS and MIM decaps in tile k by Δx_k and Δy_k, respectively in each iteration. Let $S = \sum_{i=1}^{n} S_i$ be the total violation area over all the n nodes in the supply grid. The objective function in each iteration is to minimize the total increase in the leakage power, ΔP, while maximizing the reduction in the noise violation area, S, i.e.,

$$\text{minimize} \qquad \alpha \cdot \Delta S_{scaled} + (1-\alpha) \cdot \Delta P_{scaled} \tag{4.2}$$

Here, α is a weighting parameter that sets the objective to be a convex combination of the scaled noise violation metric, ΔS_{scaled}, and the scaled leakage power, ΔP_{scaled}, where the scaling ensures that the magnitudes of the two terms are similar. The parameter ΔS is the change in S when a small amount of CMOS decap and/or MIM decap is added to each tile k. Since the amount of decap inserted in each iteration is small, this change may be computed as

$$\Delta S = \sum_{k=1}^{m'} \{(\partial S/\partial x_k) \cdot \Delta x_k + (\partial S/\partial y_k) \cdot \Delta y_k\}, \tag{4.3}$$

where m' is the number of tiles in G', $\partial S/\partial C$ is the sensitivity of S with respect to the decap $C \in \{x_k, y_k\}$, and Δx_k and Δy_k are as defined above. We note that $\partial S/\partial x_k$ and $\partial S/\partial y_k$ are nonpositive, since the violation area must decrease when decaps are added to the circuit. Therefore, minimizing ΔS, which is nonpositive, implies that we maximize the absolute reduction in S.

The leakage ΔP is calculated as $\sum_{k=1}^{m'} (a_k \cdot \Delta x_k + b_k \cdot \Delta y_k)$. In other words, it is the weighted sum of the increase in leakage due to the newly added decaps Δx_k and

Δy_k. The weights a_k and b_k are given by

$$a_k = \frac{LD_{CMOS}}{CD_{CMOS}} \cdot \phi(T_k) \tag{4.4}$$

$$b_k = \frac{LD_{MIM}}{CD_{MIM}} \cdot \phi(T_k) \tag{4.5}$$

Here, LD_{CMOS}, LD_{MIM}, CD_{CMOS}, and CD_{MIM} are, respectively, the leakage densities of CMOS and MIM decaps, and the capacitance densities of CMOS and MIM decaps, and T_k is the average temperature in the tile k. The ratio $\frac{\Delta x_k}{CD_{CMOS}}$ provides the area of the added decap, which when multiplied by LD_{CMOS} determines the corresponding leakage. The penalty term $\phi(T_k) = T_k^2 \cdot exp(\mu/T_k^2)$ captures the effect of temperature on each leakage term, where μ is a constant negative number [72]. A higher temperature T_k corresponds to a larger $\phi(T_k)$, which means that the increase in leakage in tile k is controlled more strictly.

Considering that ΔS and ΔP may have different orders of magnitude, to better control the coefficients of the objective function, we scale them to ΔS_{scaled} and ΔP_{scaled} respectively. We normalize $\partial S/\partial x_k$ and $\partial S/\partial y_k$, scaling them by $\max\{|\partial S/\partial x_k|, |\partial S/\partial y_k|\}$, so that $|\partial S/\partial x_k|$, $|\partial S/\partial y_k|$ lie in $[0, 1]$. Similarly, a_k and b_k are also scaled by the factor $\max\{a_k, b_k\}$ so that they lie in $[0, 1]$.

The weighting parameter α is dynamically adjusted: at the beginning of the optimization, it is likely that CMOS decap resources are freely available and distributed over the whole chip area, but the leakage power cost is large if we use CMOS decaps to eliminate the noise. Therefore, we choose to use small α (in the range of 0.1 to 0.2) to control the increase of leakage power, and prefer to use MIM decaps at this stage. As the optimization proceeds, since more of the noise violations are eliminated and more regions become congested after inserting MIM decaps, we increase α to use more CMOS decap to help eliminate the remaining noise. At each iteration, we track n_{vio}, the number of grids that contain violating nodes, and n_{mim}, the number of grids where MIM decaps are inserted; if n_{mim}/n_{vio} is less than 5%, then we increase α by 0.1, unless it is already equal to 1.0.

4.2.2 Constraints

1. *Congestion constraints.* Since the MIM decaps inserted between metal layers may become potential routing blockages, it is necessary to impose a constraint that restricts the deterioration of congestion with MIM decap insertion. This constraint is written as:

   $$\Delta Cong_k \leq \gamma \cdot Cong_k \tag{4.6}$$

   where $Cong_k$ is the current congestion value in tile k, $\Delta Cong_k$ is the change of the congestion in tile k in the current iteration, and γ is a bounding parameter, which is empirically set to be 0.03 to 0.05 in our experiments.

   Since each iteration imposes only a small change in the inserted decaps, it is reasonable to formulate $\Delta Cong_k$ as a linear function of the inserted MIM decaps $\Delta Cong_k = \sum_{i \in R_k}(c_i \cdot \Delta y_i)$, where the set R_k and the justification for this term are described in detail in Section 4.3.

2. *Decap resource constraints.* For a tile k, the amount of CMOS decap that can be used is limited by its available white space, and the amount of MIM decap is restricted by its capacity. If C^k_{CMOS} and C^k_{MIM} are the current maximum allocatable amount of CMOS and MIM decaps in tile k, then the decap resource constraints for tile k can be formulated as:

   $$\begin{aligned} 0 \quad &\leq \quad \Delta x_k \quad \leq \quad min\{\Delta_{CMOS}, C^k_{CMOS}\} && (4.7) \\ 0 \quad &\leq \quad \Delta y_k \quad \leq \quad min\{\Delta_{MIM}, C^k_{MIM}\} && (4.8) \end{aligned}$$

   where Δ_{CMOS} and Δ_{MIM} are upper bounds that are chosen to control the amount of CMOS and MIM decaps inserted in each iteration.

Equations (4.2)-(4.8) together formulate a linear programming problem.

4.3 Congestion Analysis And Linear Congestion Model

We estimate the routing congestion for decap optimization in 3D circuits using a probabilistic method, similar to [73], extended to 3D. However, any other congestion predictor could be used to replace this estimator with relatively few modifications, leaving the overall methodology unchanged. Given a placed 3D netlist, the core area is discretized using a 3D mesh, and the congestion in each tile of this mesh is estimated. For the purposes of our algorithm, the congestion in the Z direction is the most important: since the uppermost two layers primarily consist of supply/clock wires rather than signal wires within a single tier, MIM decaps primarily affect signal routes in the Z direction. However, other terms in the objective function can act to provide disincentives to large area capacitors which would create significant bottlenecks to power/clock wires in the X, Y, and Z directions as well.

The initial congestion map for the circuit is thus calculated, and is predicated on the assumption that there are no blockages in the region. However, in case the design uses IP blocks that impose blockages for decaps, this information may easily be incorporated into the congestion estimator. When a MIM decap is inserted, it results in a blockage and causes a perturbation in the congestion values. We model this change in the congestion in a tile cell, assuming a small perturbation as a linear function. We now describe the procedure used to calculate this linear function using the initial congestion map.

Let R_k be a set of tile cells (including k) within a specified Manhattan radius, $maxDist$, of a tile cell k. We assume that the size of R_k is bounded by a small number, reflecting the fact that we operate under small perturbations that do not cause widespread congestion changes far away from k. For each tile cell $i \in R_k$, let W_i be the current number of routes in tile cell i, and let $CurCap_i$ and $NewCap_i$ be the current and new routing capacities in tile cell i after the insertion of a MIM decap.

Let ΔW_i be the number of routes in the tile cell i to be redistributed. The redistribution process proceeds as follows after a small additional MIM decap, Δy_i, is inserted in tile cell i. If W_i is smaller than the current capacity, $CurCap_i$, then none of the routes in tile i need to be redistributed but the congestion values are updated to reflect the reduction in the capacity. Otherwise, it is necessary for routes in tile i to be redistributed. The number of routes to be moved out of tile i, to neighboring tile cells, is calculated as:

$$\Delta W_i = W_i \times \frac{CurCap_i - NewCap_i}{CurCap_i} \tag{4.9}$$

The redistribution depends on the Manhattan distance of a cell from i. For a tile cell k

that is at a distance d from cell i ($k \neq i$), the number of routes added is computed as:

$$\begin{aligned} \Delta W_{k,i} &= \frac{1}{4d} \times \frac{\omega}{d} \times \Delta W_i && (4.10) \\ \text{where} \quad \omega &= \frac{4}{\sum_{j=1}^{maxDist} (1/j^2)} && (4.11) \end{aligned}$$

The term $\frac{\omega}{d}$ captures the fact that the number of routes added to a cell varies inversely with its distance d from cell i, and these are equally distributed among the $4d$ cells that lie at a Manhattan distance of d from i. The role of the factor, ω, is to ensure that the total number of routes redistributed equals ΔW_i. In our experiments, the value of $maxDist$ is set to be 1/3 of the smaller of the number of tile cells in X and Y directions.

We then calculate $\Delta Cong_{k,i}$, the increase in congestion in tile cell k caused by $\Delta W_{k,i}$, as $\Delta Cong_{k,i} = \Delta W_{k,i} / CurCap_k = c_i \cdot \Delta y_i$ ($k \neq i$). This leads to the following linear approximation

$$\begin{aligned} \Delta Cong_k &= c_k \cdot \Delta y_k \quad + \sum_{i \in R_k, i \neq k} \Delta Cong_{k,i} \\ &= \sum_{i \in R_k} (c_i \cdot \Delta y_i) && (4.12) \end{aligned}$$

where $c_k \cdot \Delta y_k$ is the congestion increase caused by the MIM decap Δy_k added to tile k.

4.4 Sequence-of-Linear-Program Based Solution

We use an iterative flow to solve the decap allocation problem. In each iteration we allocate a relatively small amount of decap to the current circuit, for two reasons. Firstly, the decap allocation problem is highly nonlinear, and this iterative approach permits us to control the optimization process by solving a sequence of linear programs, one in each iteration. Secondly, it avoids the excessive allocation of decaps that could invalidate the approximate linear model of congestion and violation area used in our algorithm: these models are predicated on the assumption of small perturbations.

The overall optimization flow can be formulated as follows:

1. Initial setup steps: solving the input 3D power grid, determining the set of nodes that violate the voltage specifications and computing the noise violation metric,

building the coarser grid G' as described in Section 4.2, generating the temperature map for the circuit using 3D thermal analysis, and evaluating $\phi(T_k)$ in each tile k of G'.

2. If violation node set is empty, then stop. Otherwise, for each tile k that contains at least one node that violates the voltage specification, select one observation node N_k. The node N_k is chosen to be the node i with the maximum violation area, S_i, in tile k.

3. For each tile that contains an observation node N_k, calculate $\partial S/\partial C_{N_k}$, the derivative of the total violation area S with respect to the decap C_{N_k} added at N_k using the adjoint analysis method.

4. For each tile k, calculate $\Delta Cong_k = \sum_{j \in R_k}(c_j \cdot \Delta y_j)$ using the method described in Section 4.3.

5. Formulate the linear programming problem described in Section 4.2 and solve it.

6. Update the decap budget using the solution from LP solver. For each tile k, if the solution Δx_k or Δy_k of current iteration is not zero, then we insert corresponding Δx_k CMOS decap or Δy_k MIM decap to tile k. Next, we update the current maximum allocatable amount of decap resource C^k_{CMOS} or C^k_{MIM} in tile k correspondingly.

7. Solve the circuit using the updated decap allocation, and update the set of violating nodes.

8. Update the current total violation area S.

4.5 Experimental Results

The overall 3D power grid optimization flow has been written using Tcl, and the 3D power grid analyzer and the congestion and leakage aware decap allocation algorithm are implemented in C++. All experiments are performed on an Intel Pentinum 4 CPU 2.8GHz Linux machine with 1G memory running Redhat Linux 2.6.9.

The 3D placement tool in [74] is first applied to generate the 3D layouts from the IBM-PLACE benchmarks using four tiers. Next, we scale all the layouts to the $90nm$ technology node. Since the time-varying current sources, which model the behavior of each functional unit, are not originally available in these benchmarks, we use a method similar to [65] to generate the waveforms in each circuit. Six layers of regularly distributed power grid are generated for each 2D tier of a 3D circuit when building the 3D power grid. The supply voltage is set to be 1.2V and the voltage drop threshold is chosen to be 0.12V in each of the experiments. The capacitance densities for CMOS and MIM decaps are, respectively, set to be $17.3fF/\mu m^2$ (the oxide thickness is assumed to be $20\AA$) and $8.0fF/\mu m^2$. The leakage density of a CMOS decap is set to be $6.5 \times 10^{-5}mA/\mu m^2$, which is obtained from the simulation of a CMOS decap using PTM model [75]. For all of our experiments, the leakage density of the MIM decap is sufficiently small that it can be neglected.

4.5.1 Comparison Of Optimization Efficiency

Table 4.1 lists the parameters of the benchmarks used in our experiments. The circuit ibm123 is the combination of three ibm benchmarks: ibm01, ibm02 and ibm03.

Table 4.1: Parameters Of Benchmarks

Circuit	#Nodes	Worst voltage droop (V)	#Violation nodes	Violation Area S (V · ns)	Power Density ($\times 10^7$ W/m^2)	Temperature (°C)
ibm123	18634	0.135	3330	13.739	[0, 1.27]	[22.5, 88.9]
ibm05	12026	0.122	1359	72.260	[0, 1.33]	[26.2, 84.7]
ibm08	17030	0.125	3191	41.305	[0, 1.29]	[25.9, 74.3]
ibm10	29262	0.159	5935	91.286	[0, 1.25]	[26.1, 84.9]
ibm18	75042	0.163	6392	108.649	[0, 2.28]	[29.7, 92.3]

Table 4.2 lists the results of decap optimization in three different cases.

First, only CMOS decaps are used: in this case, it is not possible to add enough CMOS decaps to eliminate the the violation area **S** (see column 3) for any of the five circuits. However, we list the results for the best available solution that minimizes this metric, showing the final number of violating nodes (**VNs**) that fail to meet the constraints, the corresponding violation area (**S**), the total leakage current of the CMOS decaps (**Lkg**), the total amount of CMOS decap allocated (**CMOS**), the total number

Table 4.2: Comparison Of Optimization Efficiency

Ckt	CMOS only						MIM only					CMOS + MIM						
	VNs	S (V·ns)	Lkg (mA)	CMOS (pF)	#Iter	Time (s)	maxC (%)	avgC (%)	MIM (pF)	#Iter	Time (s)	Lkg (mA)	maxC (%)	avgC (%)	CMOS (pF)	MIM (pF)	#Iter	Time (s)
ibm123	375	0.024	2.0	543	28	141	15.8	3.9	607	7	59	1.0	8.4	1.7	271	345	5	49
ibm05	33	0.050	1.7	462	7	28	19.7	1.7	550	23	111	1.4	0.0	1.2	360	178	23	114
ibm08	38	0.011	1.1	302	19	88	30.6	1.5	768	24	134	0.5	0.0	1.0	145	622	22	121
ibm10	371	0.184	1.5	408	15	123	10.6	5.9	511	11	186	0.8	4.5	2.6	220	296	4	135
ibm18	157	0.082	2.5	673	16	450	39.5	5.3	812	9	339	1.3	7.0	3.7	344	472	9	331

of iterations required by the optimizer (**#Iter**), and the total CPU time (**Time**).

Next, only MIM decaps are used: in this case, the violation area is completely eliminated by our procedure. Considering that the allocated MIM decaps will affect the routing congestion, we list the following results: the percentage increase in maximum and average Z-direction routing congestion after optimization (**maxC**, **avgC**), the total amount of MIM decap allocated (**MIM**), as well as the total number of iterations (**#Iter**) and the total CPU time (**Time**) for this case. Since MIM decaps have much smaller leakage density than CMOS decaps, for all practical purposes, their leakage is zero and is not shown in the table.

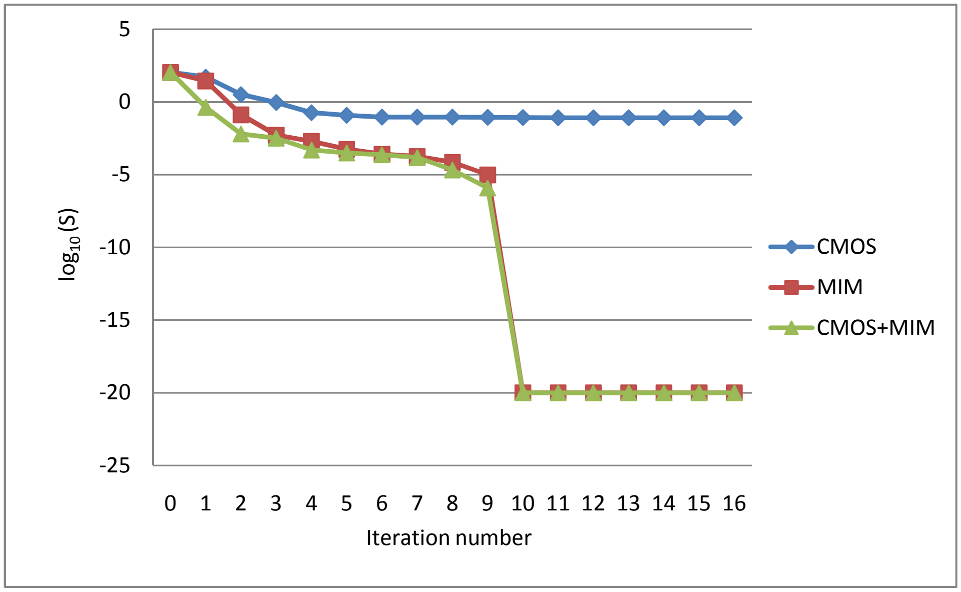
Finally, when both CMOS and MIM decaps are used, again, the violation area is completely eliminated. We list the total leakage current of the CMOS decaps (**Lkg**), the percentage increase in maximum and average Z-direction routing congestion after optimization (**maxC**, **avgC**), the total amount of CMOS decap allocated (**CMOS**), the total amount of MIM decap allocated (**MIM**), the total number of iterations (**#Iter**) and the total CPU time (**Time**).

From Table 4.2 we can see that for each of the five circuits, the violation area cannot be eliminated through the use of CMOS decaps only. This is due to the fact that the amount of CMOS decap that can be added in a circuit is limited by the available area of white space; moreover, for decaps to be effective, it is important for sufficient white space to be available near the area where the voltage constraints are violated. Placing decaps far away from the voltage violation area is of little help in alleviating noise violations. Therefore, unless we disturb the current placement or enlarge the chip size to make more white space available near the violation area, it is not possible to completely eliminate these violations.

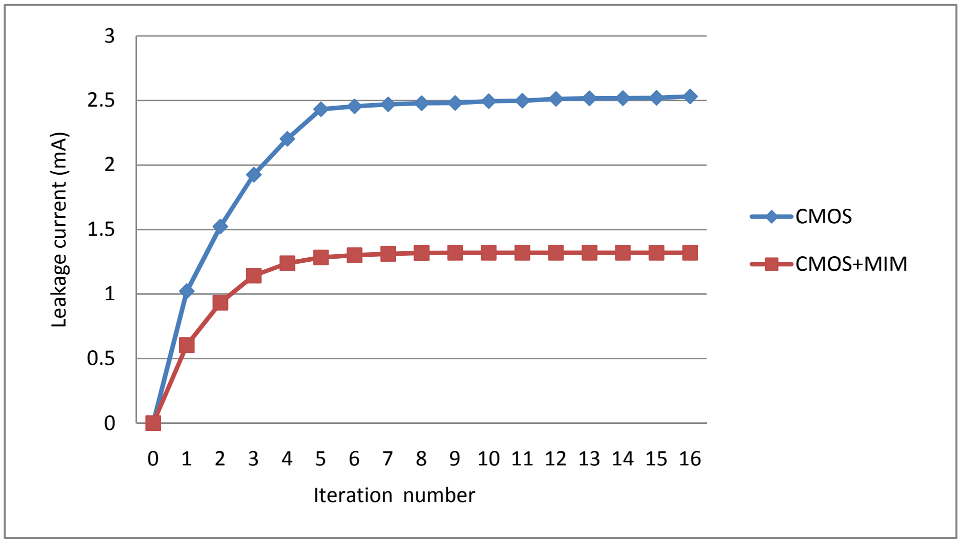
The introduction of MIM decaps can effectively eliminate the voltage violations and greatly reduce the decap leakage, at the cost of worsened routing congestion. Table 4.2 shows that the use of MIM decaps alone leads to severe congestion problems. Comparing the results of using MIM and CMOS decaps individually with using them together, it can be seen that replacing part of the MIM decaps with CMOS decaps can obtain a better tradeoff between congestion and leakage, while effectively eliminating voltage violations.

Comparing the total decap values for the MIM only and the CMOS+MIM cases, we can see that the decap values are similar (the values for CMOS-only are significantly different, since the constraints are not met in this case). The slight difference is attributable to approximations in linearizing the cost function in each iteration: specifically, in each iteration of our decap budgeting algorithm, an approximate formula, $\Delta S = (\partial S/\partial C) \cdot \Delta C$, is used to estimate the effect of added decap on the violation area, and this holds only when ΔC is small enough. In other words, in order to make the linear model more accurate, a smaller ΔC should be used, implying that the upper bounds for CMOS and MIM decaps in each iteration should be set to be very low (see Section 4.2.2). This may lead to an increase in the number of iterations, impairing the computational efficiency of our approach. In our experiments, we found that a good balance between efficiency and accuracy can be obtained when Δ_{CMOS} and Δ_{MIM} are chosen to be in the region $[0.5pF, 1.0pF]$.

Figure 4.2 shows how the total violation area and total leakage current of circuit ibm18 change as the iterative process progresses. It can be seen that the CMOS-only case cannot bring the violation area down beyond some threshold, while the MIM-only and the CMOS+MIM methods are both successful (note that the extremely low violation value of approximately 10^{-20} is essentially zero). Figure 4.2 shows that the total violation area decreases rapidly in the first 5 iterations, and most of the violations are eliminated after this stage (Note that the y-axis in this figure is on a log scale). The reason is as follows: most of the violations of the power nodes are relatively easily resolved by inserting a small amount of decap. Although the violation area of these nodes is individually small, their sum, taken over a large number of nodes, is large. Eliminating these "easy" violation at the beginning of the iterative process cause the violation area to decrease rapidly at first. Beyond this point, a relatively small number of "hard" violation nodes remain, and the change in the violation area is harder to view



(a)



(b)

Figure 4.2: Change in the total (a) noise violation area, and (b) leakage current, over each iteration.

on the scale of this graph, but is definitely visible at a magnified scale. For the same reason, most of the white space resources that are effective in reducing noise violations are consumed in early iterations, resulting in a fast initial increase in the leakage power. The leakage component of the objective function implies that MIM decaps are preferred over CMOS decaps when both are available, and when the insertion of MIM decaps does not significantly affect congestion.

4.5.2 Effect Of Power Grid Density

In this section, we further investigate how power grid density affects the results of decap budgeting provided by our algorithm. The circuit ibm123 with a size of $2480\mu m \times 2000\mu m$ was selected, and three power grids with different densities were built. In **Case1**, the power pitches in both the x and y directions, for the lowest two metal layers in each 2D tier, are set to be the cell row height. In **Case2**, the power pitch in the y direction in these layers is set to be half of the cell row height, while that in the x direction is set to be the cell row height, and in **Case3** the power pitches in both the x and y directions in these layers are set to be half of the cell row height. In all three cases, the power pitches for the higher metal layers, as well as the number of interlayer vias connecting adjacent 3D tiers, are set proportionately.

Our decap optimization algorithm, using both CMOS and MIM decaps, was then used to individually optimize the power grids in all three cases. The results are shown in Table 4.3.

Table 4.3: Optimization Results Of Different Power Grid Densities

Cases	Power Grid Density	#Nodes	#Vio. Nodes	Worst voltage droop (V)	Vio. Area (V·ns)	Decap (fF)	Lkg (mA)	maxC (%)	avgC (%)	#Iter	Time (s)
Case1	Nominal	18634	3330 (17.87 %)	0.135	13.739	616234	1.0182	8.35	1.69	5	48.7
Case2	Denser	36433	4210 (11.56 %)	0.126	2.615	436972	0.6755	27.14	4.01	2	48.8
Case3	Densest	72114	4671 (6.48 %)	0.124	1.482	237234	0.3208	58.41	7.62	1	51.3

From the table, we can see that:

- First, a denser grid helps to reduce the voltage droop in a circuit. When we increase the power grid density, both the worst-case voltage droop and violation area will be reduced (see columns 4 and 5).

- Second, a denser grid implies a larger number of grid nodes, resulting in larger cost for transient analysis and adjoint sensitivity analysis. Therefore, it takes more time to solve the problem in each iteration. On the other hand, the total number of iterations decreases because the violation area in the circuit is reduced. Therefore, we can see from Table 4.3 that the total CPU time for our algorithm increases much more slowly than the power grid size.

- Third, a denser grid implies more power connections in Z direction, and therefore a higher routing congestion, which is more sensitive to the inserted MIM decaps. This can be seen in Table 4.3: when the power grid density increases, so does the percentage increases in the maximum and average congestion values.

4.5.3 Comparison of Power Grid Performance between 2D and 3D circuits

In this section, we compare the performance of power grids in 2D and 3D circuits. In order to do so, we generate a pair of 2D circuits and a 3D circuit, all with the same footprint. The placement of these circuits is not necessarily optimized, but they are adequate for our power delivery experiments.

Several test circuits were generated by applying the 3D placement tool in [74] to find a four-tier 3D placement of the circuit ibm08. The 4 tiers of the circuit are then flattened out on the 2D plane to obtain a 2D circuit, A_1, by placing tier0 at the lower left, tier1 at the lower right, tier2 at the upper right and tier3 at the upper left. A different 2D circuit, A_2, was obtained from the same 3D placement result by changing the order in which the tiers were placed: this time, tier2 is placed at the lower left, tier3 at the lower right, tier0 at the upper right and tier1 at the upper left. Finally, we stacked circuit A_2 on A_1 to build a separate 3D circuit, B.

By construction, the circuits A_1, A_2, and B all have the same footprint, and circuits A_1 and A_2 have the same average current densities. The first three rows of Table 4.4 show the characteristics of these three circuits. As expected, it can be seen that before optimization, the 3D circuit has a significantly worse voltage drop, and a larger fraction of the total number of nodes experience noise violations.

Next, we applied our MIM+CMOS algorithm to optimize these three circuits, and the results are summarized in the last four columns of the table. It can be seen that

Table 4.4: Comparison of power grid performance between 2D and 3D circuits

Ckt	#Node	V_{dd} (V)	Worst voltage droop (V)	#Vio. nodes	Vio. area S (V · ns)	Lkg (mA)	Total decap (pF)	maxC (%)	avgC (%)
A_1	16529	1.2	0.1348	1368 (8.28%)	3.728	1.909	795.3	0.0	0.12
A_2	16529	1.2	0.1354	1537 (9.30%)	4.292	2.083	876.1	0.0	0.13
B	33033	1.2	0.1868	11421 (34.57%)	229.327	23.411	8210.9	0.0	0.23
A'_2	16529	1.2	0.1693	3110 (18.82%)	54.632	6.635	2641.9	0.0	0.32
B'	33033	1.2	0.2098	15827 (47.91%)	445.633	32.238	11130.7	0.0	0.38
A''_2	16529	1.2	0.1016	0 (0 %)	0	-	-	-	-
B''	33033	1.2	0.1638	6287 (19.03%)	92.913	10.680	3532.5	0.0	0.10

circuit B requires about 5× the amount of decaps as the combined sum for A_1 and A_2 to meet the voltage specifications, which results in a larger amount of leakage. This motivates the need for using MIM decaps, to control the leakage.

Next, we consider two scenarios where the current distribution on the upper layer is increased by 25% (circuit A'_2) or decreased by 25% (circuit A''_2). This mimics the fact that circuit designers may choose to use more memory on one layer than the other. Correspondingly, these circuits are stacked on A_1 to obtain 3D circuits B' and B'', respectively. These results are shown in the lower half of Table 4.4. As expected, both A'_2 and B' require more decaps than A_2 and B, respectively. Moreover, There is no voltage violation for circuit A''_2, although B'' does see violations that must be fixed, at a cost lower than that for B.

4.6 Conclusion

We have proposed an efficient decap allocation algorithm to optimize 3D power supply network using both MIM and CMOS decaps. MIM decaps have the desirable properties of high capacitance density and low leakage density, and can be a good complement to the on chip SiO_2-based CMOS decap. Our algorithm uses 3D congestion analysis and a linear congestion model, as well as linearized noise models based on adjoint sensitivity analysis, to guide the decap allocation among CMOS and MIM decaps. Experimental results show that power grid noise can be more effectively optimized using both MIM and CMOS decaps, with lower leakage power and low routing congestion costs.

Chapter 5

Exploration of On-Chip Switched-Capacitor DC-DC Converter for Multicore Processors Using a Distributed Power Delivery Network

In this and next chapters, we present techniques for building and optimizing the power delivery network for multicore processors, that go beyond the decap optimization explored in Chapter 4. This is related to a recent technical advance that makes it possible to integrate on-chip voltage regulator structures, which are very effective in maintaining supply voltage levels.

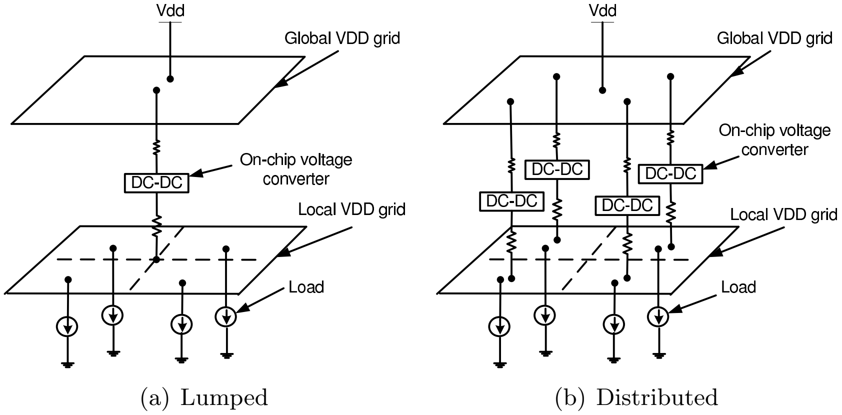
In this chapter, we explore the design of on-chip SC converters, using an accurate power grid simulator. Results show that distributed design of SC converters can reduce the IR drop by up to 74% compared to the lumped design, with improved supply voltage. We also demonstrate the usage of SC converters for multi-domain power supply.

5.1 Introduction

The roadmap for future multicore-based computing shows more and more processor cores placed on the same die to build CMPs. CMPs provide the ability to perform multiple tasks in parallel. However, the power demands of various cores on the same die can be different, and can change with time, depending on the applications that they may run. Dynamic voltage scaling (DVS) is one of the most effective means to achieve energy-efficient design in CMPs. The varying power demands of all cores can be best met if DVS is supported by providing multiple independent on-chip power supplies: this can support per-core or per-cluster (where multiple cores are driven by the same supply) power management in CMPs.

A voltage regulator is an essential component of the power delivery network. Most DVS systems are based on off-chip voltage regulators driving on-chip power grids, which comes at the cost of additional complexity and area, since voltage regulators are built traditionally in board-level with large inductors or capacitors. The costs and sizes of such bulky modules severely limit their use for multiple power domain regulation. To enable per-cluster or per-core DVS, it is essential to develop fully integrated on-chip DC-DC converters for each power domain, which can significantly improve load regulation and eliminate load-transient spikes caused by inductances from package and global power grid [76, 77].

The key challenge associated with realizing such on-chip integrated converters is the difficulty in achieving high efficiency at the high power densities required by high-performance CMPs. Historically, on-chip DC-DC converters are limited to low power applications [78, 79] due primarily to the lack of dense, high-quality-factor energy storage elements. In typical CMOS processes, on-die capacitors have significantly higher Q and energy density and lower cost than on-die inductors, leading to several recent efforts in exploring fully integrated SC DC-DC converters [79–81]. In [81], the authors have demonstrated the application of embedded deep trench capacitors in a SC DC-DC voltage converter to provide an on-chip energy storage device of extreme density (∼200nF/mm^2, current density of 2.3A/mm^2), high efficiency (90%) and minimal parasitic losses.



(a) Lumped (b) Distributed

Figure 5.1: Lumped vs. Distributed on-chip DC-DC converters.

Prior work has not adequately studied the layout implications of on-chip power supply design. It is well known that power delivery is most efficient if the power sources are close to the utilization points (it is for this reason that decoupling capacitors – which deliver power based on stored charge – are placed close to large noise sources). In this work, we explore the application of on-chip SC DC-DC converters in the context of CMPs. When integrating SC converters into on-chip power delivery network, we can built them in either lumped or distributed form, as shown in Figure 5.1. For the lumped case, a large central converter delivers power to all the blocks in the DVS cluster or the whole chip. In contrast, for the distributed case, several smaller converters can be distributed across the chip and each load can absorb current from the nearby converter. Although an independent closed-loop control unit is needed for each distributed converter [78], its benefits are significant. First, since the load current is typically at the granularity of Amps in CMPs [82], distributed converters can significantly reduce the voltage droop seen by the local loads by providing more localized power distribution. Second, distributed design of converters provides the flexibility to support multiple power deliveries, and we can apply DVS to each local converter to achieve better power management.

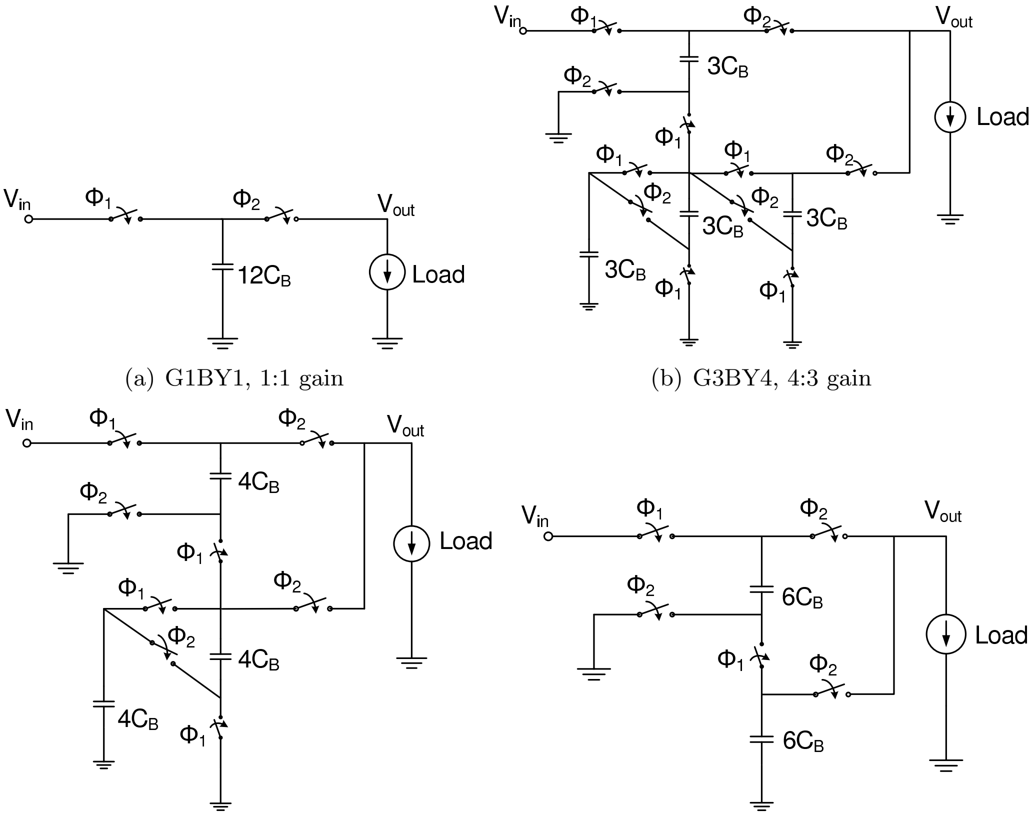
Existing design tools do not provide adequate support for analyzing multicore power grids. Therefore, we develop an accurate on-chip power grid simulator which incorporates on-chip SC DC-DC converters and supports multiple power domains. We then quantitatively compare the lumped and distributed designs of on-chip SC converters using realistic current profiles from CMP applications. We also demonstrate the application of SC converters for multi-domain power supply.

5.2 Switched-Capacitor DC-DC Converter

A SC DC-DC converter (also known as a charge pump) is a network of charge-transfer capacitors (also called pumping capacitors) and switches that operates in two or more phases, converting an input voltage V_{in} to an output voltage of V_{out}. If V_{out} is higher than V_{in}, the conversion is called a "step-up" conversion. Vice versa, if V_{out} is lower than V_{in}, the conversion is called a "step-down" conversion. In this work we focus on step-down conversion.

A representative SC DC-DC converter operates in two non-overlapping phases: a charging phase ϕ_1 and a discharging phase ϕ_2 (in reality, many more phases are used to control ripple: in this paper, we use 16 phases – but the essential idea is the same as for two phases). During phase ϕ_1 a group of capacitors in the network are connected to the input to get charged, while in phase ϕ_2 this group of capacitors are connected to the output to discharge. There are several different ways to configure the connection of capacitors in each phase, and each configuration has its own characteristics. In this work, we explore the simple"Series-Parallel" configuration. Figure 5.2 shows four different kinds of series-parallel step-down SC DC-DC converters as proposed in [78]. This method uses the same total capacitance of $12C_B$ and and provides multiple output voltage levels from the same converter block through various series-parallel reconfigurations of this total capacitance.

For example, consider configuration G1BY2, with 2:1 gain (conversion ratio). Two capacitors each with capacitance of $6C_B$ and five switches are connected in a network, and the switches are controlled by two signals ϕ_1 and ϕ_2 (Figure 5.2(d)). In charging phase ϕ_1, ϕ_1 turns ON two switches connecting the two charge-transfer capacitors in series (Figure 5.3(a)). Since both capacitors have the same value of capacitance $6C_B$, each will be charged to $V_{in}/2$ if enough time is provided for the capacitors to be fully charged. In the second phase ϕ_2, three other switches turn ON, while the ones controlled by ϕ_1 turn OFF (Figure 5.3(b)). This will connect both capacitors in parallel with the output load, resulting in an output voltage $V_{out} = V_{in}/2$. As current starts to flow into



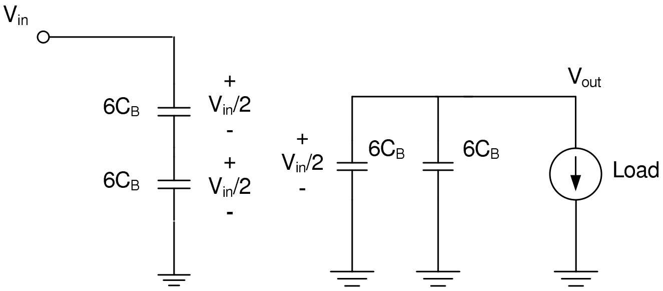
(a) G1BY1, 1:1 gain

(b) G3BY4, 4:3 gain

(c) G2BY3, 3:2 gain

(d) G1BY2, 2:1 gain

Figure 5.2: Configurations of SC DC-DC converters with different gains.



(a) Charging phase

(b) Discharging phase

Figure 5.3: Equivalent circuit in charging and discharging phases for G1BY2.

the load, the charge stored in the capacitors will deplete and the output voltage will drop to $V_o = V_{in}/2 - \Delta V$ at the end of this stage before it is recharged in the next phase.

The power that such an SW DC-DC can deliver is

$$P_L = (\alpha \cdot C_B \cdot V_{in} \cdot \Delta V) \cdot f_s \cdot \eta \tag{5.1}$$

where α is a coefficient determined by the particular topology, f_s is the switching speed of clock signals ϕ_1 and ϕ_2, and η is the efficiency of the converter. For further details, the reader is referred to [78].

5.3 Simulation Platform

Figure 5.4 presents a detailed model of the power delivery network for the CMP.

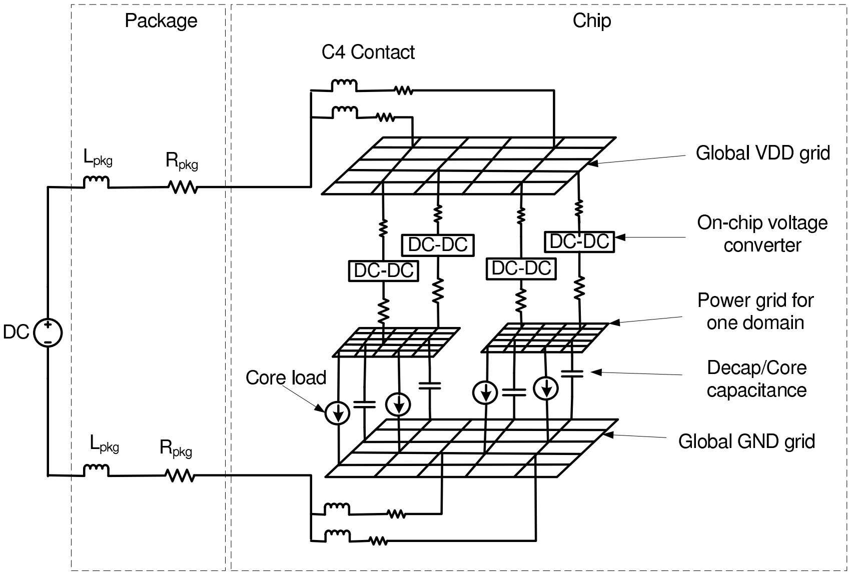


Figure 5.4: Model of power delivery network.

The package and C4 bump contacts are modeled as RL pairs. The on-board power supply is modeled as a DC voltage source. The on-chip power delivery network consists of a global VDD grid, on-chip DC-DC converters, local power grids, a global GND grid, core or decoupling capacitors and current loads. The global sparse VDD grid distributes voltage to on-chip SC converters. Each local power grid belongs to a power domain, and its voltage is controlled by the corresponding on-chip SC converters. Each power domain can have a group of SC converters. The power grids are generated according to an industrial 32nm technology.

In our work, we consider multicore applications which require multiple power delivery domains for best energy efficiency. Existing power grid simulators, which are focused on simulating a single voltage domain, are excellent for today's CMPs that use a single off-chip voltage regulator. However, they do not provide adequate support for simulating large power grid networks driven by SC DC-DC converters, incorporating factors such as the regulator efficiency under time-varying loads. Therefore, we build an accurate power grid simulator incorporating on-chip SC DC-DC converters.

We consider a test chip with four identical cores. Figure 5.5 shows the chip floorplan.

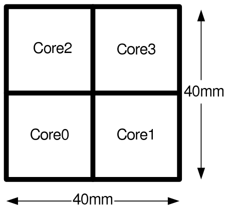
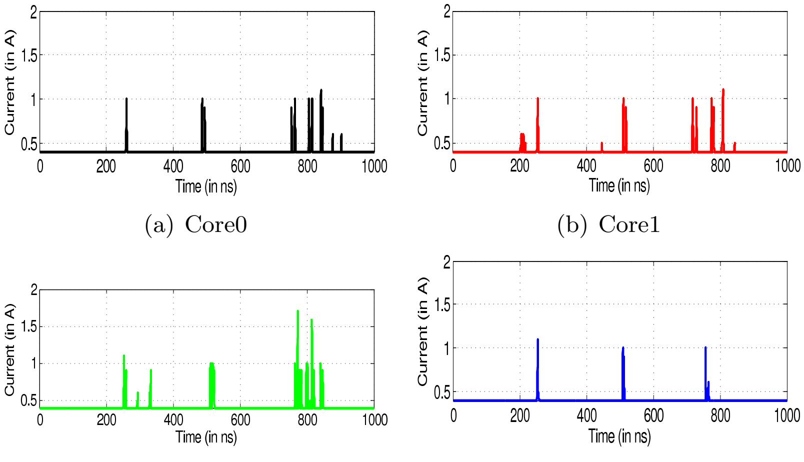


Figure 5.5: A CMP with four cores.

In our simulator, each core can be modeled as either lumped or distributed time-varying current sources. In our simulations, we model each core as a lumped current source and generate the current profiles by simulating several SPEC OMP2001 [62] workloads using an accurate full system multicore simulator GEMS [59]. We observed two typical types of current traces from these workloads: in one, which we call *trace1* and show in Figure 5.6, there are many short current pulses early in the simulation, while the other, which we call *trace2*, is of the nature shown Figure 5.7.

For the SC converters, we use the structures shown in Figure 5.2. The switches are modeled as resistors when they are turned on. As a common practice, 16-phase interleaving (within each converter, 16 cells working in parallel) is use to reduce the output ripple of the converters. The digital-capacitance-modulation scheme [79] is integrated into our simulator, which controls the amount of capacitance that takes part in the charge transfer process.



(a) Core0

(b) Core1

(c) Core2

(d) Core3

Figure 5.6: *Trace1* for four cores.

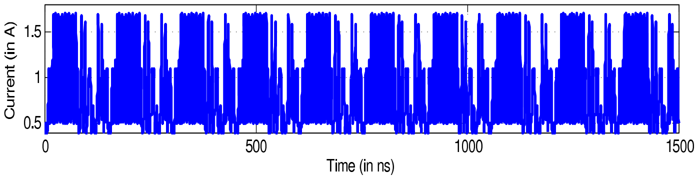


Figure 5.7: *Trace2*, the apparent periodicity is caused by a loop in the execution.

Further, we explore the choice of C_B. Depending on the current demands, a larger or smaller C_B may be used. We organize the C_B capacitors into banks so that each C_B can have four different sizes: 1X, 2X, 4X and 8X, and any C_B capacitors that are not used (e.g., for a low current demand) can be power-gated to reduce leakage. It should be noted that the maximum available C_B for the lumped and distributed designs of the SC converters are different, since distributed converters are smaller, more numerous, and must satisfy lower local power demands, they may use smaller C_B values.

The parameters for the SC converters studied in this work are summarized in Table 5.1, and the other parameters for the power grid and the CMP are listed in Table 5.2.

Table 5.1: Summary of SW DC-DC converters

Capacitance density	0.2μF$/mm^2$
Total area	30.72mm^2
Total capacitance	6.144μF
# of cells	16
Duty cycle	50%
f_s	100 Mhz
C_B	8nF for distributed, 32nF for lumped.
Switch resistance	20$m\Omega$

Table 5.2: Simulation Configuration

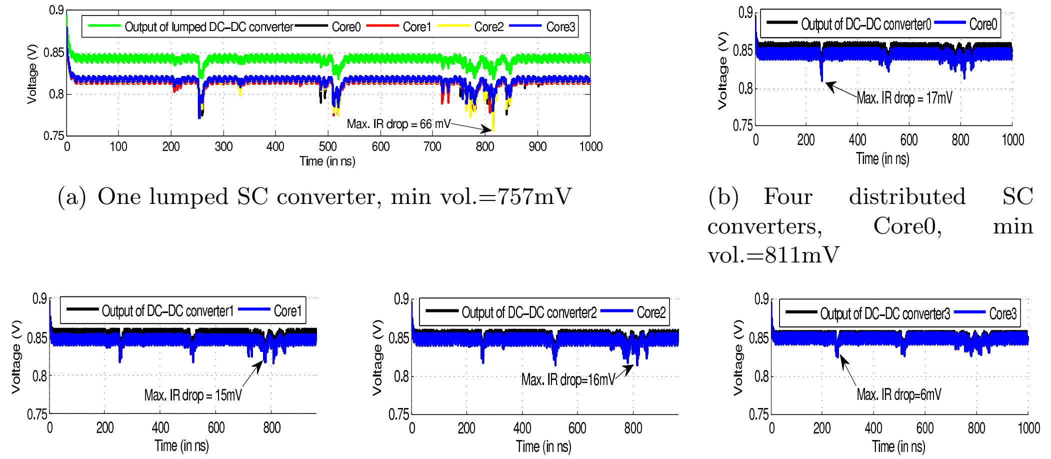
DC voltage source	Vdd=1.2V (Vcore = 0.6V~1.2V)
Package	$L_{pkg} = 15pH$, $R_{pkg} = 1m\Omega$
C4 bump	#=768, $L_{bump} = 7.2pH$, $R_{bump} = 1.5m\Omega$
Core load	capacitance=1 nF, core frequency=750Mhz

5.4 Simulation Results

5.4.1 Lumped vs. Distributed On-Chip SC DC-DC Converters

In this section, we compare the lumped and distributed designs of on-chip SC converter. For this experiment, we assume that all the four cores shown in Figure 5.5 works in one power domain, and the G3BY4 structure (Figure 5.2(b)) with 4:3 conversion ratio and a nominal Vdd of 0.9V is used to deliver power to the cores. For the lumped design, we place a single SC converter in the center of the test chip, and it delivers power to all the four cores; for the distributed design, we place four individual SC converters evenly distributed on the chip, so that each core can absorb current from its local converter. For fair comparison, the same amount of total available charge-transfer capacitances are used for the lumped and distributed cases.

We exercised these two designs by applying the two types of current traces shown in Figures 5.6 and 5.7. The *trace1* current profile can serve as the low load case, and the *trace2* for the high load case. The results are shown in Figures 5.8 and 5.9. From Figure 5.8 we can see that for a nominal voltage of 900mV and *trace1*, compared to the lumped design, the minimum voltage seen by the cores can be improved from 757mV to 811mV, and the maximum IR drop can be reduced by 74% if we go for the distributed design. The corresponding numbers for *trace2* is an improvement of minimum voltage from 637mV to 729mV and a 71% reduction of maximum IR drop, as shown in Figure 5.9.



(a) One lumped SC converter, min vol.=757mV

(b) Four distributed SC converters, Core0, min vol.=811mV

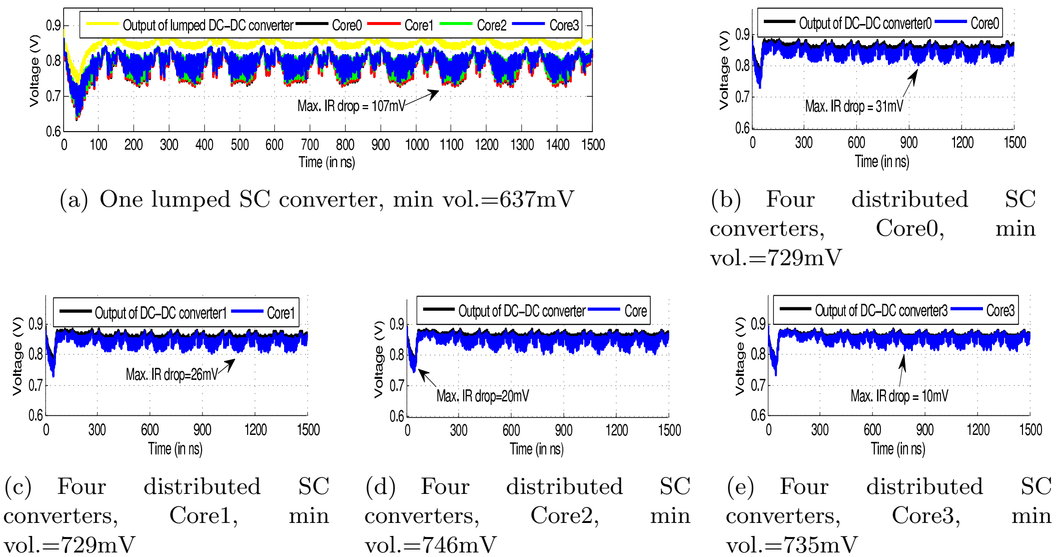
(c) Four distributed SC converters, Core1, min vol.=816mV

(d) Four distributed SC converters, Core2, min vol.=815mV

(e) Four distributed SC converters, Core3, min vol.=826mV

Figure 5.8: Comparison of lumped and distributed designs of SC converter using current profile *trace1*.

Efficiency is an important metric for SC DC-DC converters. The principle contributors to efficiency loss in a SC DC-DC converter are: conduction loss arising from charing a capacitor through a switch, loss due to parasitic capacitors, gate-drive loss



(a) One lumped SC converter, min vol.=637mV

(b) Four distributed SC converters, Core0, min vol.=729mV

(c) Four distributed SC converters, Core1, min vol.=729mV

(d) Four distributed SC converters, Core2, min vol.=746mV

(e) Four distributed SC converters, Core3, min vol.=735mV

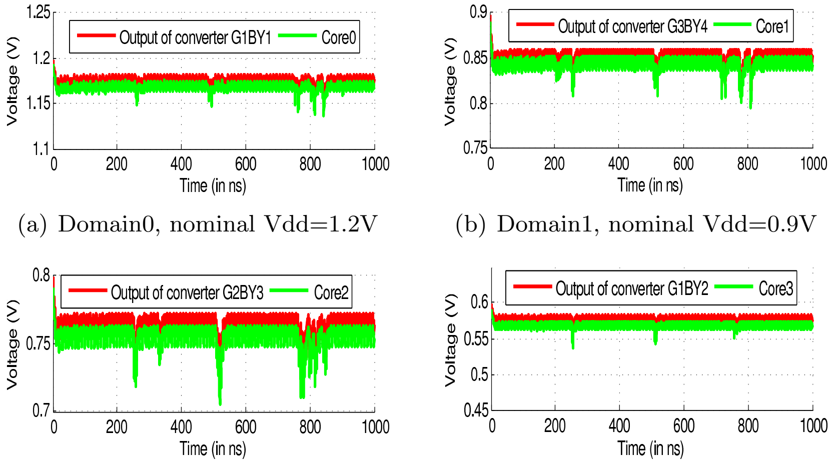
Figure 5.9: Comparison of lumped and distributed designs of SC converter using current profile *trace2*.

due to switching the gate capacitance of the charge-transfer switches, and power loss in the control circuitries [78]. Simulation results show that the parasitic capacitance of the deep trench capacitors is less than 1% of the total charge-transfer capacitance. The size of the switches is negligible compared to the cores in our test chip, so we can ignore the gate-drive loss. The loss in the control circuitry is of specific concern only when delivering ultra-low load power levels (in the magnitude of μW) [78]. Therefore, in our simulations the loss of the SC converters mainly come from the conduction loss. The measured results show that, for the lumped and distributed converters, the average efficiencies when simulating the current profiles are in the range of [92.39%,95.38%].

In summary, although the distributed design requires an independent closed-loop control unit for each individual converter, its benefits are prominent. First, in the distributed design, the cores can absorb current from local SC converters, and the current doesn't need to flow through a long conduction path from the converter to the core load as in the lumped case. Therefore, distributed design of SC converters would benefit in the sense of less IR noise since each converter can regulate its local supply voltage. Second, the distributed converters deliver much less power than the lumped one, with smaller capacitors they can respond much faster to the changes in the local core loads, which leads to smaller voltage swings as seen by the loads. Finally, distributed SC converters have the flexibility to manage the charge-transfer capacitors in fine granularity: when a local core is idle at the execution time, the corresponding local converter can power-gate its unused charge-transfer capacitors to reduce leakage power.

5.4.2 Multiple Power Deliveries Using On-Chip SC DC-DC Converters

In this section, we explore the use of on-chip SC converters for multi-domain power delivery. For the test chip shown in Figure 5.5, we design four power domains: Core0 works in domain0, served by one lumped G1BY1 converter with nominal Vdd of 1.2V, Core1 works in domain1, served by one lumped G3BY4 converter with nominal Vdd of 0.9V, Core2 works in domain2, served by one lumped G2BY3 converter with nominal Vdd of 0.8V, and Core3 works in domain3, served by one lumped G1BY2 converter with nominal Vdd of 0.6V.



(a) Domain0, nominal Vdd=1.2V

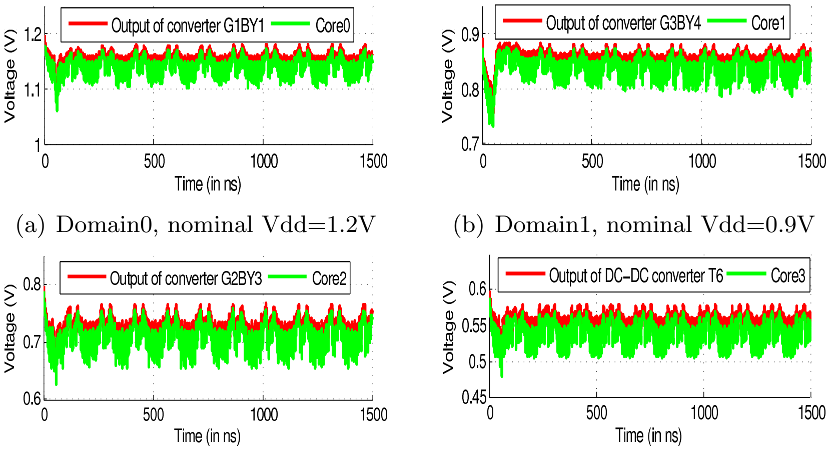
(b) Domain1, nominal Vdd=0.9V

(c) Domain2, nominal Vdd=0.8V

(d) Domain3, nominal Vdd=0.6V

Figure 5.10: Simulations results of four power domains using *trace1*.

We then exercised these four power domains by the corresponding current traces presented in Section 5.3. Figures 5.10 and 5.11 show the simulation results. We can see that all the four domains work well. In fact, given a single Vdd supply, we can further dynamically reconfigure the converter in each domain (see Figure 5.2) to deliver a wide



(a) Domain0, nominal Vdd=1.2V

(b) Domain1, nominal Vdd=0.9V

(c) Domain2, nominal Vdd=0.8V

(d) Domain3, nominal Vdd=0.6V

Figure 5.11: Simulations results of four power domains using *trace2*.

range of load voltages, therefore DVS can be applied to each domain to achieve better power management.

5.5 Conclusion

In this work, we have explored the design of on-chip SC DC-DC converters with an accurate power grid simulator. Simulation results based on realistic multicore current profiles show that distributed SC converters can reduce the IR drop by up to 74% compared to the lumped design, with improved supply voltage. We also present the idea of using SC converters for multi-domain power supply.

Chapter 6

Optimization of On-Chip Switched-Capacitor DC-DC Converters for High-Performance Applications

In this chapter, we continue to explore the optimization of SC converters. The efficiency of the power delivery system using SC converters is a major concern, but this has not been addressed at the system level in prior research. This chapter develops models for the efficiency of such a system as a function of size and layout of the SC converters, and proposes an approach to optimize the size and layout of the SC converter to minimize power loss. The efficiency of these techniques is demonstrated on both homogenous and heterogenous multicore chips.

6.1 Introduction

Figure 6.1 shows a simplified power delivery system including the global V_{dd} supply, a switched-capacitor (SC) converter to convert the input V_{dd} to required voltage supply level, a power grid to distribute the power to local core loads, and a core load. The output of the converters is V_{cvt}, but the exact voltage supply seen by the cores is

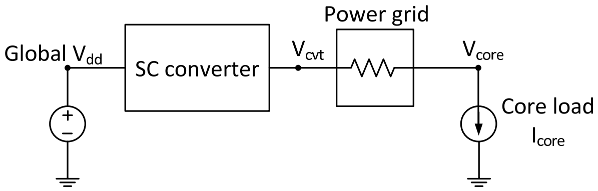


Figure 6.1: Schematic of a power delivery system.

downgraded to V_{core} due to losses such as voltage droop (e.g., due to IR drop) in the power delivery network. To overcome these losses and ensure correct core operation, the nominal V_{dd} supply of the power domain, $V_{vdd,dom}$, must be set to

$$V_{vdd,dom} = V_{vdd,core} + V_{droop} + \Delta V/2 \tag{6.1}$$

where $V_{vdd,core}$ is the minimum voltage specified at the core load, V_{droop} is the peak voltage droop between V_{cvt} and V_{core}, and ΔV is the peak-to-peak output voltage ripple of the converter. For a core that draws current I_{core}, the power supplied by the converters is:

$$P_{cvt} = I_{core} V_{vdd,dom} \tag{6.2}$$

However, the power needed by the core loads is smaller:

$$P_{core} = I_{core} V_{vdd,core} \tag{6.3}$$

The remainder of the power,

$$I_{core}(V_{droop} + \Delta V/2),$$

is wasted in various parts of the power delivery network. The efficiency of power delivery is given by:

$$\eta = \frac{\text{Power delivered to the load}}{\text{Power extracted from the input } V_{dd} \text{ supply}} \tag{6.4}$$

Prior work on optimizing on-chip capacitive DC-DC converters is very limited. The work in [83] has focused primarily on reducing wasted power *within* the internal design of the converter (i.e., entirely inside the "SC converter" box in Figure 6.1) by controlling the voltage ripple ΔV, optimizing efficiency by choosing the optimal switch width and switching frequency. Under this paradigm, the burden of optimizing the other term for the voltage droop, V_{droop}, (corresponding to the "Power grid" box in Figure 6.1) is placed on conventional means for power grid optimization, e.g., grid topology selection and wire widening. In Chapter 5, we address the problem by suggesting the use of distributed SC converters, which can significantly reduce the voltage droop seen by the local core loads by providing more localized power distribution; however, we have not looked into the efficiency optimization problem.

In this chapter, we take a novel approach to the problem and consider a more holistic optimization of the DC-DC converter at the system level. We differ from prior efforts in considering not only the internals of the converter but also its context within the system to which it delivers power. In particular, we show that by optimizing the number and layout of the converters for the power domain, it is possible to control the losses due to wasted power in the power grid and enhance the efficiency of the converter. To the best of our knowledge, this is the first work to address efficiency optimization at the system level.

The rest of this chapter is organized as follows. In Section 6.2, we present some basic principles of SC converters. This is followed, in Section 6.3, by a description of our proposed models for various components of the power loss as a function of the size and layout of the SC converters in a power delivery system based on SC converters. Next, in Section 6.4, we present the problem formulation of the efficiency optimization problem, followed by a description of our approaches for solving the problem in Section 6.6. Finally, in Section 6.7, the efficiency of our approaches is demonstrated on both homogeneous and heterogeneous multicore chips.

6.2 SC DC-DC converters

A block diagram of a general SC converter system is shown in Figure 6.2. The system consists of N_{phase} interleaving stages (a typical value of N_{phase} is 32), which reduce the ripple voltage by $1/N_{phase}$ compared to an SC converter without any interleaving.

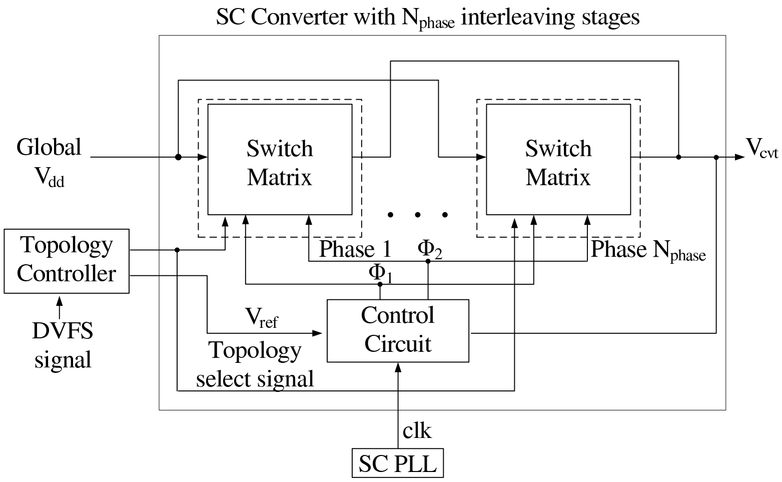


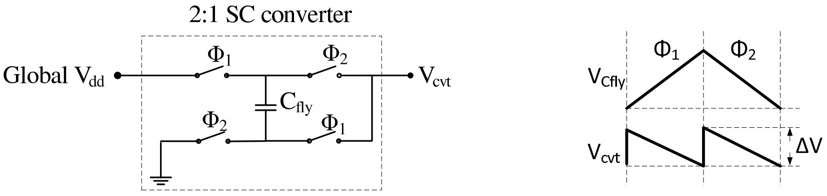
Figure 6.2: Block diagram of an SC DC-DC converter.

At the core of the system is the switch matrix, one for each phase [1]. This matrix is a reconfigurable arrangement of switches and flying capacitors that is configured in different ways by the "Topology select" signal from the topology controller. Each such configuration provides the ability to produce a different voltage conversion ratio, allowing the converter to generate one of several output voltage levels from the converter [78]: for simplicity, these details are not shown here. The conversion ratio of the converter is defined as the ratio between the input voltage, which is the external supply voltage, V_{dd}, and the desired output voltage, $V_{vdd,dom}$, which is the specification for the ideal value of V_{cvt}. The control circuit takes these inputs:

- the clock signal *clk* from a phase-locked loop (PLL)
- the reference voltage for a particular topology V_{ref}
- the feedback voltage V_{cvt} from the converter output

It generates the nonoverlapping clock signals Φ_1 and Φ_2 for the switches in the switch matrix, and may also be used to gate some of the capacitors to control the amount of capacitance that takes part in the charge transfer process [79].

A switch matrix topology is shown in Figure 6.3(a), with a 2:1 conversion ratio. Figure 6.3(b) (top) shows that during Φ_1, the flying capacitor C_{fly} is connected to the input global V_{dd} to get charged, and during Φ_2, the charge stored in C_{fly} is transferred to the load and its voltage drops by ΔV as it is discharged. This is reflected as the output voltage at the output, V_{cvt} of the converter in Figure 6.2, as shown in Figure 6.3(b) (bottom) in Φ_2 [83].



(a) (b)

Figure 6.3: (a) The topology of a 2:1 SC converter (b) Its output waveform.

Note that another switch matrix is connected to the output during Φ_1 (and is charged during Φ_2), which results in the voltage ripple observed in the V_{cvt} waveform.

Note that the signals Φ_i are generated by a relatively low-frequency clock ($f_{sw} \approx 100MHz$), which is distinct from the multi-GHz clock used by the multicore processor.

6.3 Power Loss Analysis

Efficiency is one of the key design metrics for the on-chip DC-DC converters [77, 83]. We now analyze the inefficiency and power loss in a SC converter. Our analysis is based on [1, 83, 84], as well as from conversations with designers. Some items in this section are taken from the literature, while others are freshly derived.

For each converter, let f_{sw} be the switching frequency of the converter, $C_{sw} = C_{fly} \times N_{phase}$ be the total amount of flying capacitance, and ΔV be the output ripple of the converter.

(1) Conduction loss: This corresponds to the power loss in the switches as the flying capacitors are charged. For each converter, the conduction loss is modeled as:

$$P_{cond} = M_{sw} \frac{I_{out}^2}{N_{phase}} \frac{R_{on}}{W_{sw}} \tag{6.5}$$

where M_{sw} is a constant determined by the converter topology (Table 6.1), I_{out} is the total current delivered by the converter, R_{on} is the switch resistance per unit width, and W_{sw} is the switch width.

Conversion ratio	M_{sw}	γ	M_p	M_{topo}
1:1	1	1	0	1/2
4:3	7/3	2/3	$3/8\alpha$	8/9
3:2	1	1	$1/3\alpha$	9/8
2:1	2	2	$1/4\alpha$	2

Table 6.1: M_{sw}, γ, M_p and M_{topo} for different topologies [1]. α is the ratio of the plate capacitance to its effective capacitance.

For a given topology, W_{sw} is proportional to f_{sw} and C_{sw}:

$$W_{sw} = \sigma\gamma f_{sw}\frac{C_{sw}}{N_{phase}} \tag{6.6}$$

where σ is a fitting coefficient, and γ is topology-dependent (Table 6.1). In an SC converter supporting DVFS, the switch size may be adjustable, where some of a set of parallel switches are turned on to achieve the desired switch size [84].

(2) Gate-drive loss of the switches: The switches in a converter are implemented using transistors. These transistors must be very wide in order to minimize conduction losses, and therefore the power loss in driving their gate nodes can be modeled as:

$$P_{sw} = N_{phase} \cdot N_{sw} \cdot f_{sw} \cdot (C_{gate}W_{sw}) \cdot V_{dd}^2 \tag{6.7}$$

where N_{sw} is the number of switches used in one particular topology and C_{gate} is the per-unit-width gate capacitance of the switches.

(3) Parasitic loss: This is the loss from the bottom-plate parasitic capacitance of the flying capacitors. The loss can be estimated as:

$$P_{para} = M_p f_{sw} C_{sw} V_{dd}^2 \tag{6.8}$$

where M_p is a parameter that depends on the internal structure of a topology (Table 6.1). This loss component depends on the particular type of the capacitance technology. Deep trench capacitors typically have superior efficiency compared to MIM and CMOS capacitors.

(4) The load power loss: The load power loss $I_{core}(V_{droop} + \Delta V/2)$, described in Section 6.1, can be separated into two parts:

(4a) The part determined by the voltage ripple, ΔV, is

$$P_{L1} = I_{core}\Delta V/2 \tag{6.9}$$

In each cycle, the energy a topology can deliver is given by $M_{topo}C_{sw}N_{phase}\Delta V$, where M_{topo} is determined by the topology (Table 6.1), because with the same amount of flying capacitance C_{sw}, different topologies can deliver different amount of power to the output. When switching at frequency f_{sw}, the current a converter can provide is

$$I_{out} = M_{topo} \cdot f_{sw} \cdot C_{sw} \cdot N_{phase} \cdot \Delta V \tag{6.10}$$

$$\text{i.e.,} \qquad \Delta V = \frac{I_{out}}{M_{topo}f_{sw}C_{sw}N_{phase}} \tag{6.11}$$

From Equation (6.11), we can see that with the same output current I_{out}, the voltage ripple ΔV is inversely proportional to the size of charge-transfer capacitance C_{sw}.

(4b) The power loss associated with the voltage droop, V_{droop}, is

$$P_{L2} = I_{core}V_{droop} \tag{6.12}$$

Note that the voltage droop changes as we alter the number and locations of the converters on the chip, since the distance between the converters and the utilization points (cores) changes.

(5) Control circuit: The control unit generates the nonoverlapping clock signals for the switches used in the converter. This unit includes a voltage comparator, DLL and control logic, and its total power is

$$P_{ctrl} = P_{ctrlr}N_{cvt} \tag{6.13}$$

where P_{ctrlr} is the power of each of the N_{cvt} converters used.

(6) Clock network: The power consumption of the clock network arises from the wire capacitance, the clock buffers inserted for the wires, and the clock loads, and can be expressed as:

$$\begin{aligned} P_{clock} &= f_{sw}(C_{wire} + C_{buffer} + C_{clk-load})V_{dd}^2 \\ &\quad + V_{dd}I_{leak-buffer} \\ &= \beta Length_{wire} + \lambda N_{cvt} \end{aligned} \tag{6.14}$$

where β and λ are parameters dependent on technology, $Length_{wire}$ is the total length of wires in the clock network.

(7) Clock sources: The clock source is implemented as a simple PLL with relaxed frequency (≈ 100MHz) and jitter (less than tens of ps) requirements compared to the main PLL for the on-chip circuit. Thus, the power consumption of the clock source is $P_{clksrc} = P_{PLL}$, where P_{PLL} is the power consumption of one PLL [85].

(8) Topology controller: This generates the signals that provide DVFS directives to reconfigure the topology in each converter to set the conversion ratio that provides the desired voltage output level. The topology controller is a small combinational logic block and its power consumption is in the order of μW, which is ignored here.

6.4 Optimization Formulation

In the scenario studied here, it is safe to assume that the switching frequency f_{sw} and interleaving stages N_{phase} are fixed for the converters. Based on the analysis in Section 6.3, the components of power loss can be divided into four categories.

The first component, which depends on the parameters of the converter, is the power consumption of the conduction loss/gate-drive loss of the switches/parasitic loss/part of load loss P_{L1}, and is determined by the C_{sw} and the global V_{dd}, as:

$$P_1 = P_{cond} + P_{sw} + P_{para} + P_{L1} \tag{6.15}$$

For each converter, we can change the total flying capacitance, C_{sw}, to tune the voltage ripple ΔV, according to Equation (6.11). A larger C_{sw} results in smaller ΔV, and can therefore reduce the *load power* P_{L1} (Equation (6.9)) and *switch conduction loss* P_{cond} (Equations (6.5) and (6.6)). On the other hand, the *gate switching loss* P_{sw} (Equations (6.6) and (6.7)) and *parasitic loss* P_{para} (Equation (6.8)) increase with C_{sw}.

An optimal value of C_{sw} balances these conflicts.

The second and third components are, respectively, the power consumption of part of load loss P_{L2}, and the sum of the power loss in the control circuit and clock network.

$$\begin{aligned} P_2 &= P_{L2} && (6.16) \\ P_3 &= P_{ctrl} + P_{clock} && (6.17) \end{aligned}$$

Both P_2 and P_3 are determined by the number and layout of the converters. Changing the granularity of the capacitance through more fine-grained distributed converters placed over the chip (as opposed to a single centralized converter) can help reduce the voltage droop seen by the core loads, therefore reduce the loss P_{L2} [86]. However, using a larger number of converters implies higher cost for the hardware implementation due to higher losses in the control circuit and clock network. Therefore, it is necessary to explore the number and layout of the DC-DC converters to determine an optimum.

The last component, corresponding to the loss of the clock sources and topology controllers is fixed and is given by

$$P_4 = P_{clksrc} + P_{topctrlr} \tag{6.18}$$

At the system level, the efficiency of the power delivery system is

$$\eta = \frac{P_{core}}{P_{core} + P_1 + P_2 + P_3 + P_4} \tag{6.19}$$

where P_{core} is defined in Equation (6.3). To increase the efficiency, we minimize the sum of P_1 through P_4, which constitute the power wasted during power delivery. Further, since P_4 is a fixed quantity, to improve the overall efficiency of the power delivery system using SC converters, we should optimize the objective function:

$$\text{minimize} \quad P_1 + P_2 + P_3 \tag{6.20}$$

The variables in the optimization problem are

- the number of converters used, N_{cvt},
- the capacitance of each used converters C_{sw}, and

- the locations of the converters.

The optimization is subject to the following constraints:

1. The supply voltage at each core load must meet a lower bound:

$$V_{core} \geq V_{vdd,core} \tag{6.21}$$

2. Since the voltage ripple constraint must limit $\Delta V \leq \Delta V_{max}$, Equation (6.11) provides a bound on C_{sw}:

$$C_{sw} \geq \frac{I_{out}}{M_{topo} f_{sw} N_{phase} \Delta V_{max}} \tag{6.22}$$

3. To control the capacitance resource used, we require that:

$$\sum C_{sw} \leq C_{max} = C_{unit} Area_{max} \tag{6.23}$$

   where C_{unit} is the capacitance density, and $Area_{max}$ is the maximum available area for the converters.

6.5 MINLP Formulation

Figure 6.4 presents a schematic of the on-chip power delivery network for a multicore processor. The on-chip power delivery network consists of a global V_{dd} supply, on-chip DC-DC converters, the power grid, and core loads. The voltage supplied to the power grid controlled by a set of on-chip SC converters, which can be placed at a list of predefined candidate locations on the chip.

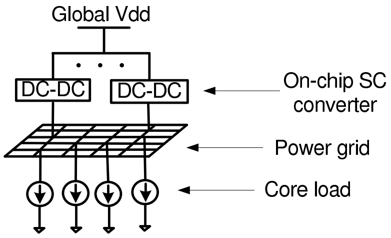


Figure 6.4: Model of power delivery network.

In the following sections, we show that the optimization problem in Section 6.4 can be formulated as a mixed-integer nonlinear program problem (MINLP), by introducing 0–1 integer variables z_is, with $z_i = 1$ denoting that a converter is placed at candidate location i. We first macromodel the power grid in Section 6.5, and then present the complete MINLP formulation in Section 6.5.

Macromodeling of the power grid

The power grid may have millions of nodes, but we are only interested in OBS, the selected n observation nodes of the core loads, and Src, the m predefined candidate connection nodes for the SC converters. Therefore, we build a macromodel whose ports are these $n + m$ nodes, and abstract away all of the other nodes in the network using the macromodeling approach [87]. Therefore, Figure 6.4 is transformed to the model shown in Figure 6.5.

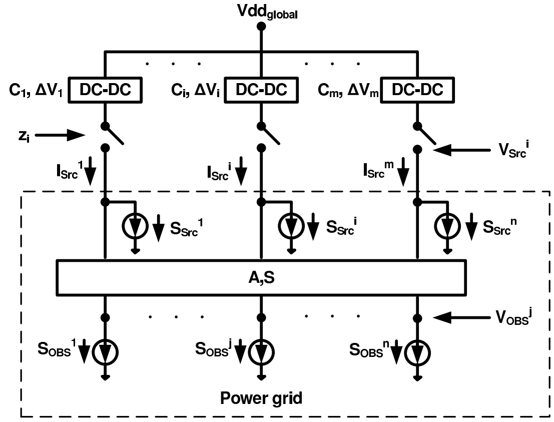


Figure 6.5: Macromodel of the power delivery network.

The DC analysis of a V_{dd} power grid is formulated as:

$$Gv = i \tag{6.24}$$

where G is the conductance matrix for the interconnected resistors, v is the vector of node voltages, and i is the vector of current loads. The equations for the power grid are given as

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} -J_1 + I \\ -J_2 \end{bmatrix} \tag{6.25}$$

where U and V are voltages of the ports and internal nodes, J_1 and J_2 are current sources connected at ports and internal nodes, and I is the vector of current flowing into the macromodel through the ports. The macromodel of the power grid including only the port nodes (cores' accessing nodes OBS and the candidate nodes for the converters Src) is given by

$$I = AU + S \tag{6.26}$$

where $A = G_{11} - G_{12}G_{22}^{-1}G_{21}$, and $S = J_1 - G_{12}G_{22}^{-1}J_2$. By partitioning the ports into sets Src and OBS, this can be rewritten as

$$\begin{bmatrix} I_{Src} \\ I_{OBS} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} V_{Src} \\ V_{OBS} \end{bmatrix} + \begin{bmatrix} S_{Src} \\ S_{OBS} \end{bmatrix} \tag{6.27}$$

where (I_{Src}, V_{src}) and (I_{OBS}, V_{OBS}) are the (current,voltage) values at the Src and OBS ports. Since $I_{OBS} = 0$, we have:

$$V_{OBS} = T \cdot V_{Src} + B \tag{6.28}$$

where $T = -A_{22}^{-1}A_{21}$, and $B = -A_{22}^{-1}S_{OBS}$. Further,

$$I_{Src} = A_{11}V_{Src} + A_{12}V_{OBS} + S_{Src} = A'V_{Src} + S'_{src} \tag{6.29}$$

where $A' = A_{11} + A_{12}T$ and $S'_{src} = S_{Src} + A_{12}B$.

From Equations (6.28) and (6.29) we can see that the current vector of the Src ports I_{Src} and voltage vector of the OBS ports V_{OBS} are linear functions of the voltage vector of the Src ports V_{Src}.

MINLP Formulation

Using the macromodel shown in Figure 6.5, the optimization problem described in Section 6.4 is equivalent to finding the optimal z_i assignments, and for each used converter

i (with $z_i = 1$), determining its size C_i and voltage ripple ΔV_i.

We rewrite P_1 (Equation (6.15)), the power loss associated with the converter and the global V_{dd} supply, as:

$$P_1 = \sum_{i=1}^{m} \left(e_1 e_3 I^i_{Src} \Delta V_i + e_2 V^2_{vdd,dom} C_i \right) \tag{6.30}$$

where

$$\begin{aligned} e_1 &= \left(\frac{1}{2N_{phase} M_{topo}} + \frac{M_{sw} R_{on}}{\sigma\gamma} \right) \frac{1}{f_{sw}} \\ e_2 &= f_{sw} \left(N_{sw} C_{gate} f_{sw} \sigma\gamma + M_p \right) \cdot ratio^2_{cvt} \\ e_3 &= N_{phase} M_{topo} f_{sw} \end{aligned}$$

Here $ratio_{cvt}$ is the conversion ratio of the converters.

Using Equation (6.28), P_2, the power loss in the grid, and P_3 are:

$$\begin{aligned} P_2 &= \underbrace{\sum_{i=1}^{m} (V^i_{Src}(I^i_{Src} - S^i_{Src}))}_{\text{Power supplied to the macromdel}} - \underbrace{\sum_{j=1}^{n} (V^j_{OBS} S^j_{OBS})}_{\text{Power delivered from the macromodel}} \\ &= \sum_{i=1}^{m} \left(V^i_{Src}(I^i_{Src} - S'^i_{Src}) \right) - \sum_{j=1}^{n} (B^j S^j_{OBS}) \qquad (6.31) \\ P_3 &= P_{ctrl} + P_{clock} = c \cdot \sum_{i=1}^{m} z_i \qquad (6.32) \end{aligned}$$

where c is penalty weight for control circuit and clock network, $V_{vdd,dom}$, V^i_{Src}, I^i_{Src}, C_i, ΔV_i are the continuous variables and z_is are the 0–1 integer variables in the optimization problem.

Then we can transform the optimization problem defined in Section 6.4 into a MINLP formulation as

$$\begin{aligned} \text{min.} \quad P_1 + P_2 + P_3 &= \sum_{i=1}^{m} \left(e_1 e_3 I^i_{Src} \Delta V_i + e_2 V^2_{vdd,dom} C_i \right) \\ &+ \sum_{i=1}^{m} \left(V^i_{Src}(I^i_{Src} - S'^i_{Src}) \right) - \sum_{j=1}^{n} (B^j S^j_{OBS}) + c \sum_{i=1}^{m} z_i \qquad (6.33) \end{aligned}$$

subject to

$\forall j \in$ OBS:

$$V_{OBS}^{j} = \sum_{i=1}^{m} (T_{ji} \cdot V_{Src}^{i}) + B^{j} \geq V_{th}^{j} \tag{6.34}$$

$\forall i \in$ Src:

$$I_{Src}^{i} = \sum_{k=1}^{m} (A'_{ik} \cdot V_{Src}^{k}) + S'^{i}_{Src} \tag{6.35}$$

$$I_{src}^{i} = e_3 \cdot \Delta V_i \cdot C_i \tag{6.36}$$

$$0 \leq I_{Src}^{i} \leq M \cdot z_i \tag{6.37}$$

$$\frac{I_{Src}^{i}}{e_3 \cdot \Delta V_{max}} \leq C_i \leq M \cdot z_i \tag{6.38}$$

$$0 < \Delta V_i \leq \Delta V_{max} \tag{6.39}$$

$$\sum_{i=1}^{m} C_i \leq C_{max} \tag{6.40}$$

$$V_{Src}^{i} + \frac{\Delta V_i}{2} \leq V_{vdd,dom} \tag{6.41}$$

Here, V_{th}^{j} is the minimum required voltage at the observation nodes of each core, and M is a large positive number.

Constraints (6.34) are transformed from Equation (6.21), to specify the minimum voltage for each core load. Constraints (6.35) are from Equation (6.29), and Constraints (6.36) from Equation (6.11). Constraints (6.37) are structured to ensure that the current I_{src}^{i} is zero when no converter connected to candidate port i, while Constraints (6.38) ensure that converter size C_i is zero when I_{src}^{i} is zero, both through the use of M. Constraints (6.39) and (6.40) are from Equations (6.22) and (6.23), and Constraints (6.41) set the bound for the Vdd supply.

We can observe that there are nonlinear (actually non-convex) terms in the objective function (6.33) and constraints (6.36) are also nonlinear. Therefore, the above optimization problem is a MINLP.

6.6 Heuristic Approaches

As stated in [88], “MINLP problems are difficult to solve precisely, because they combine all the difficulties of both of their subclasses: the combinatorial nature of mixed integer programs (MIP) and the difficulty in solving nonconvex (and even convex) nonlinear programs (NLP). Because subclasses MIP and NLP are among the class of theoretically difficult problems (NP-complete), so it is not surprising that solving MINLP a challenging and daring venture.”

Therefore, in our work we explore heuristic approaches to solve the optimization problem. For the objective function in Equation (6.33),

- $P_2 + P_3$ is determined by the number/layout of the converters
- P_1 is determined by the converter design, i.e, the size of converters C_i, and $V_{vdd,dom}$, the V_{dd} supply. From Equation (6.1) we can see that $V_{vdd,dom}$ is determined by the voltage droop in the power grid and the ripple in the converters.

Therefore, we may optimize the power loss in two steps. We first optimize $P_2 + P_3$, the power in the distribution network, by finding the optimal number and layout of the converters. We present two heuristic approaches in Section 6.6.2 for this step. Next, we optimize P_1 to determine the optimal size of each used converter C_i, which is presented in Section 6.6.3.

6.6.1 An approximation for the voltage ripple

We introduce the approximation that all converters have the same voltage ripple. In other words,

$$\Delta V_i = \Delta V \ \forall \ i \text{ such that } z_i = 1.$$

The impact of this assumption is that by Equation (6.36), the current delivered by a converter i is proportional to its capacitance C_i, which is a reasonable assumption.

We justify this approximation as follows. In Equation (6.30), let P_1^i be the contribution of the i^{th} converter to P_1. If $z_i = 1$,

$$P_1^i = e_1 e_3 I_{Src}^i \Delta V_i + e_2 V_{vdd,dom}^2 C_i \tag{6.42}$$

According to Equation (6.36), P_1^i is equivalent to

$$P_1^i = e_1 \frac{(I_{Src}^i)^2}{C_i} + e_2 V_{vdd,dom}^2 C_i \tag{6.43}$$

If we minimize P_1^i locally by setting $\partial P_1^i / \partial C_i = 0$, we get

$$C_i = \frac{I_{Src}^i}{V_{vdd,dom}} \sqrt{\frac{e_1}{e_2}} \tag{6.44}$$

Therefore, according to Equation (6.36) we can see that

$$\Delta V_i = \frac{I_{Src}^i}{e_3 C_i} = \frac{V_{vdd,dom}}{e_3} \sqrt{\frac{e_2}{e_1}} \tag{6.45}$$

Since e_1, e_2, and e_3 are constants, and $V_{vdd,dom}$ is common to all the converters, ΔV_is can be assumed to be the same among the used converters if they are locally optimized. Therefore, in the following discussion, we assume $\Delta V_i = \Delta V$ for each used converter.

If all C_is were free variables, allowed to take any value, this would not be an approximation. However, according to Equation (6.40), the C_is are not unconstrained, therefore this is an approximation.

6.6.2 Optimizing Converter Number/Layout

As stated earlier, the number and layout of the converters also affects the efficiency of the power delivery system. Distributing the converters with finer granularity and optimized layout over the chip can help improve the efficiency loss by reducing the voltage droop seen by the local core loads, when placing the converters closer to the utilization points. However, there is an overhead associated with the power loss in the control units and clock network.

How significant is the converter area?

At this point, it is useful to consider some technology numbers to determine the area overheads of the SC converters. To compute this, we assume that the SC converters are fabricated using deep-trench capacitors. In [81], the reported power density of deep-trench capacitors is 200nF/mm^2. A typical core has the current of $\sim$ 1A. According to Equation (6.10), if we use a 2:1 converter (with $M_{topo} = 2$) to deliver this amount of current with ripple $\Delta V = 5$mV, $N_{phase} = 32$ and $f_{sw} = 100$Mhz, then the required amount of capacitance is 31.25nF, which transforms to 0.156mm^2. Considering that the typical size of a core is of several mm^2, we may ignore the area effect of the converters when optimizing the layout of the converters.

MILP-based Approach

In this section, we present a MILP-based heuristic to optimize the number and layout of the converters. We proceed under the assumption that for each used converter, $\Delta V_i = \Delta V$, and define

$$V_{vdd,local} = V_{vdd,dom} - \Delta V/2 \tag{6.46}$$

From Equation (6.41) we can see that

$$V^i_{Src} \le V_{vdd,local} \tag{6.47}$$

The power loss due to voltage droop, P_2, shown in Equation (6.31), can be relaxed as

$$\begin{aligned} P_2 &= \sum_{i=1}^{m}(V^i_{Src} I^i_{Src}) - \sum_{i=1}^{m}(S'^i_{Src} V^i_{Src}) - \sum_{j=1}^{n}(B^j S^j_{OBS}) \\ &\le V_{vdd,local}\sum_{i=1}^{m} I^i_{Src} - \sum_{i=1}^{m}(S'^i_{Src} V^i_{Src}) - \sum_{j=1}^{n}(B^j S^j_{OBS}) \end{aligned} \tag{6.48}$$

Essentially, since $I^i_{src} = 0$ when $z_i = 0$, the substitution in the first term means that $V^i_{Src} = V_{vdd,local}$. In the above expression, $\sum_{i=1}^{m} I^i_{Src}$ is the total current delivered to the cores, and therefore, a constant. We can see that by relaxation we can transform the nonlinear cost function P_2 to be linear.

In fact, in our experiments using all approaches, we find that V^i_{Src} is nearly equal for every converter i, so that (6.47) is in practice an equality, confirming the validity of the minimizing the relaxed P_2.

Since $\sum_{j=1}^{n}(B^j S^j_{OBS})$ is a constant, it is unchanged under any optimization. Then the relaxed power loss (P_2 + P_3) can be minimized by solving the following MILP problem:

$$\text{min.} \quad V_{vdd,local}\sum_{i=1}^{m} I_{Src}^{i} - \sum_{i=1}^{m}(S_{Src}^{\prime i}V_{Src}^{i}) + c\sum_{i=1}^{m} z_i \tag{6.49}$$

subject to the linear constraints in Equations (6.34), (6.37) and (6.47).

Note that I_{Src}^{i} is substituted with V_{Src}^{i} according to Equation (6.35), so this MILP formulation has m 0-1 integer variables (z_is), $m+1$ continuous variables ($V_{vdd,local}$ and V_{Src}^{i}s) and $3m+n$ constraints.

Greedy Approach

Considering that MILP can be expensive for a large number of integer variable z_is, we propose a greedy approach to reduce the run-time complexity of solving the optimization problem with a large set of candidate locations for the converters. The idea is to explore different granularity of converters: from one converter for each core, to a single lumped converter for all the cores.

For a chip with n cores, the inputs of the greedy approach include

1. A list of cores $\Re = \{C_0, \ldots, C_l\}$. Core C_i has peak current I_i and minimum required voltage supply V_{vdd,C_i},
2. A adjacency graph G_0 representing the neighbor relationships among the l cores; if a layout is provided instead, this information can be generated using Voronoi diagrams.
3. A list of all candidate locations $\Psi = \{\psi_1, \ldots, \psi_m\}$ for the converters on the chip (Figure 6.7 shows part of the candidate set that are used by the converters).

The edge weight w_{ij} of an edge between vertices i and j in the adjacency graph is calculated as the increase in the power loss from combining two converters V_i and V_j into a single converter, V_{ij}. This quantity is the total change in the power loss $P_2 + P_3$, which includes:

1. the change in power loss from voltage droop [Equations (6.1), (6.2) and (6.12)]

$$\Delta P_{L2} = \Delta V_{vdd,dom} \cdot \textstyle\sum_{i=1}^{l} I_i$$

2. the change in power loss from the control circuit ΔP_{ctrl} [Equation (6.13)]
3. the change in power loss from the clock network ΔP_{clock} [Equation (6.14)]

i.e, $w_{ij} = \Delta P_{L2} + \Delta P_{ctrl} + \Delta P_{clock}$ where ΔP_{L2} is non-negative because voltage droop tends to increase with fewer converters, $\Delta P_{ctrl} = -P_{ctrlr}$ because the number of converters is reduce by one after combining two converters into one, and ΔP_{clock} is determined by the locations of the converters V_i, V_j and V_{ij}. Note that w_{ij} can be negative in our approach.

Our approach to optimizing the converter design is iterative in nature, and the overall scheme is illustrated in the left half of Figure 6.6. We begin with a design with one individual converter for each core. The top right box in Figure 6.6 shows an example of the given adjacency graph G_0 for the l cores. In G_0, each node V_i represents the converter for core C_i.

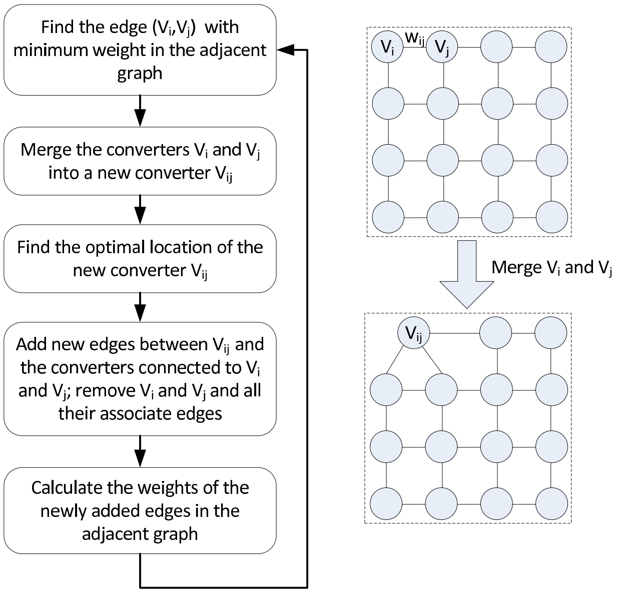

Find the edge (V_i,V_j) with minimum weight in the adjacent graph
Merge the converters Vi and Vj into a new converter Vij
Find the optimal location of the new converter Vij
Add new edges between Vij and the converters connected to Vi and Vj; remove Vi and Vj and all their associate edges
Calculate the weights of the newly added edges in the adjacent graph
Vi
wij
Vj
Merge Vi and Vj
Vij

Figure 6.6: Outline of the proposed approach to explore different granularity of converters.

The principle behind our method is to begin with the adjacency graph, allowing each core to have its own converter. Then we contract edges in the graph to reduce the number of converters by merging the adjacent converters. Starting from a given adjacency graph G_0 with l converters, at each iteration we greedily merge the neighboring converters V_i and V_j with minimum edge weight w_{ij}, so as to minimize the possible increase of power loss at the next level of converter granularity. When merging two neighboring converters V_i and V_j, two nodes in the adjacent graph is merged into one new node, and the weights of the edges between this new node edge and its neighbours are updated as stated earlier.

We compute the optimal location, as described in the next paragraph, for the combined converter V_{ij}, and then update the adjacency graph. With l cores, our approach will repeat the merging process $l-1$ times to evaluate all possible levels of converter granularity.

We select the location of a converter V_i from the set of candidate locations Ψ to minimize the nominal output voltage of the converters, minus the voltage ripple part [Equation (6.1)], i.e.,

$$V_{vdd,local} = V_{vdd,dom} - \frac{\Delta V}{2} = \max_{i \in \{1,\ldots,l\}} (Vdd_{C_i} + V_{droop,i}) \tag{6.50}$$

where $V_{droop,i}$ is the voltage drop at core C_i. When evaluating each candidate location, the voltage droop of each core can be obtained from the simulation of the power grid. However, consider that the power grid is typically costly to simulate, to speed up the evaluation process, we assume that the conduction resistance between a core C_i and its converter V_j is linearly proportional to their distance $Dist(C_i, V_j)$, i.e., $V_{droop,C_i} = I_i \cdot R_{unit} \cdot Dist(C_i, V_j)$, where R_{unit} is the unit-distance resistance of the power grid. However, the voltage droop for our final results are validated using a accurate circuit simulator.

6.6.3 Optimization of Converter Size

After determining the number and layout of converters using the heuristic approaches in Section 6.6.2, the second step is to determine C_i for each converter i by optimizing P_1.

Let $I_{total} = \sum_{i=1}^{m} I_{Src}^i$ and $C_{total} = \sum_{i=1}^{m} C_i$, then from Equation (6.45) we can see that

$$\Delta V = \frac{I_{Src}^i}{e_3 C_i} = \frac{I_{total}}{e_3 C_{total}} \tag{6.51}$$

so to minimize the power loss P_1 in Equation (6.30) is equivalent to minimizing

$$\begin{aligned} P_1 &= e_1 e_3 \Delta V I_{total} + e_2 V_{vdd,dom}^2 C_{total} \\ &= e_1 I_{total}^2 \frac{1}{C_{total}} + e_2 V_{vdd,dom}^2 C_{total} \end{aligned} \tag{6.52}$$

Using Equation (6.46), Equation (6.52) can be further transformed to

$$\begin{aligned} P_1 &= e_1 I_{total}^2 \frac{1}{C_{total}} + e_2 (V_{vdd,local} + \Delta V/2)^2 C_{total} \\ &= e_2 V_{vdd,local}^2 C_{total} + I_{total}^2 (e_1 + \frac{e_2}{4e_3^2}) \frac{1}{C_{total}} \\ &\quad + \frac{e_2}{e_3} V_{vdd,local} I_{total} \end{aligned} \tag{6.53}$$

where I_{total} is a constant, and $V_{vdd,local}$ can be found after solving the optimization problem in Section 6.6.2. The constraints for the above problem is given by Equation (6.40) and

$$C_{min} = \frac{I_{total}}{e_3 \Delta V_{max}}, \tag{6.54}$$

which is derived from Equations (6.39) and (6.51).

Note that P_1 is a *convex* function of C_{total}. It is easily determined that the optimal solution to the unconstrained problem defined in Equation (6.53) is given by:

$$C_0 = \frac{I_{total}}{V_{vdd,local}} \sqrt{\frac{e_1 + \frac{e_2}{4e_3^2}}{e_2}} \tag{6.55}$$

However, this value of C_0 may fall outside the bounding constraints (6.40). If so, from the convexity of the objective function, we can conclude that the optimum must be at the extreme point of the allowable C_{total} interval that is closer to C_0.

Next, the optimal size of C_{total} for the converters, C_{opt}, is

$$C_{opt} = \begin{cases} C_{min} & \text{if } C_0 < C_{min} \\ C_0 & \text{if } C_{min} \le C_0 \le C_{max} \\ C_{max} & \text{if } C_0 > C_{max} \end{cases} \tag{6.56}$$

Then we can calculate the voltage ripple ΔV according to Equation (6.51) using C_{opt}, and the optimal size of each used converter C_i can be calculated by Equation (6.51) because I^i_{Src} is known after solving the optimization problem in Section 6.6.2.

6.7 Experimental Results

Our heuristic approaches described in Section 6.6 are implemented in C++. The MILP problem is solved using CPLEX [89].

6.7.1 Test Cases

Our approaches were exercised on two chips, one of which is a homogeneous multicore while the other is a heterogenous multicore processor. The configuration of each chip is described below:

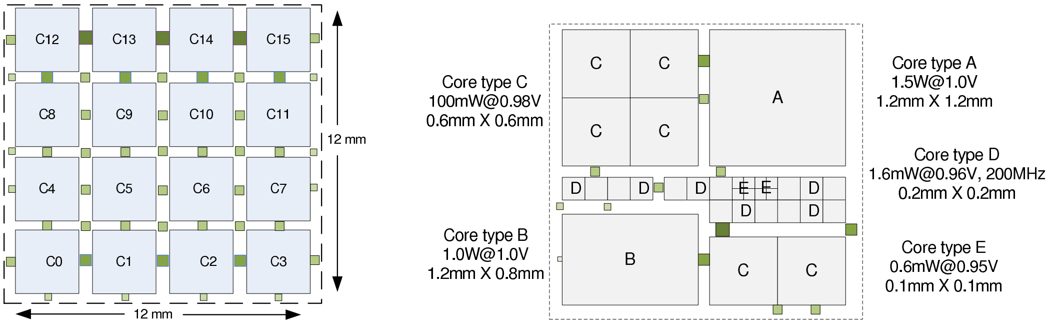


Figure 6.7: Two test cases with 16 homogeneous cores (left) and 32 heterogeneous cores (right)

Homogeneous Chip: Our homogeneous test case consists of a chip with one power domain of 16 identical cores, as shown in Figure 6.7 (left), which follows the tile-based design for multicore chip [82]. Each core consists of a CPU, L1 I/D cache and L2 cache with area ratio of 2:1:2. The core is $3 \times 3mm^2$ with a peak current of 1A@0.6V. In our simulations, we model the current ratio among CPU, L1 cache and L2 cache inside each core using guidelines consistent with [7].

Heterogeneous Chip: We also consider a heterogeneous test case consisting of a set of ARM Cortex cores [90]. Simpler versions of such heterogeneous cores are already on the market today [91]. This test case has one power domain of 32 cores as shown in

Figure 6.7 (right). Core types A through E are, respectively, the A9, A8, A5, M4, and M0 cores.

	Homogeneous case	Heterogeneous case
Conversion ratio	2:1	3:2
Total core current I_{core}	16A	3.14A
Max output ripple ΔV_{max}	10mV	20mV
Area$_{max}$	28.8mm^2	1.056mm^2
C_{max}	5.76 μF	0.21μF
Number of switches N_{sw}	4	7
M_{topo} (Refer to Equation (6.11))	2	9/8

Table 6.2: Configurations of the two chips.

Switching frequency f_{sw}	100Mhz
Interleaving stages N_{phase}	32
Capacitance density C_{unit}	200nF/mm^2
Unit-length gate capacitance C_{gate}	3fF/μm
Unit-width switch resistance R_{on}	130$\Omega \cdot \mu$m
Control circuit power P_{ctrlr}	4.0mW
Parasitic parameter α	0.1%
Clock power parameter β	1.27mW/mm
Clock power parameter λ	4.95μW
Switch width coefficient σ	512μm/(μF·MHz)

Table 6.3: Global configuration parameters.

Table 6.2 shows the parameters for these two chips, and Table 6.3 lists the main parameters used in our design of converters. These parameters are evaluated under the 32nm technology node using data from the published literature and PTM [75] files. We assume that the available area for the converters is up to 20% of the total core area.

6.7.2 Comparison of Heuristic Approaches

We have presented two heuristic approaches for the optimization of the number and layout of the converters in Section 6.6.2, followed by the optimization of converter size using a closed-form solution. The first heuristic approach (refer to Section 6.6.2) *Heuristic-MILP* formulates the optimization as a MILP problem, and the second heuristic approach *Greedy* in Section 6.6.2 uses *greedy* strategy to explore the number and layout of converters at different levels of granularity. We compare these two approaches with a manual design approach, which evenly distributes the converters over the chip at different levels of granularity with total number of converters set to $2^k, k = 0, 1, 2, \ldots, \lfloor \log_2^m \rfloor$, where m is the numbers of candidate locations for the converters

Table 6.4: Comparison Of Optimization Efficiency, without limitation on # converters

Chip	m	n	Manual						Greedy							Heuristic-MILP						
			#cvt	P1	P2	P3	Total	η	#cvt	P1	P2	P3	Total	η	CPU	#cvt	P1	P2	P3	Total	η	CPU
Homo16	56	208	32	763	574	128	1465	86.1	36	706	389	144	1239	87.6	5.9	47	705	283	188	1176	88.1	370.1
Hete32	76	203	16	160	277	64	501	86.1	11	157	184	44	385	88.9	1.7	13	157	141	52	350	90.1	362.7
Average					1		1				0.67		0.81					0.50		0.75		

Table 6.4 shows the results of these approaches. Columns 2–3 show m, the numbers of candidate locations for the converters, and n, the number of observation nodes for the cores. Columns 4–9 show the results of manual design, columns 10–16 give the results of the greedy scheme discussed in Section 6.6.2, and columns 17–23 show the results of the heuristic approach presented in Section 6.6.2. For each approach, we list the total number of converters used, the total power loss (refer to Equation (6.20)) and its breakdown, P_1, P_2, and P_3, in mW. We also show η, the system-level efficiency of the power delivery system, and CPU, the runtime of these two heuristic approaches in seconds (on a 64-bit 2.5GHz Intel Quad-core platform).

On average, compared to the manual design, the greedy approach can reduce P_2 (the power loss due to voltage droop) by 33%, and total power loss by 19% with higher system-level efficiency. The heuristic approach based on MILP can reduce P_2 by about 50% and total power loss by 25%. The system-level efficiency is improved from 86.1% to 88.1% for the homogeneous chip, and from 86.1% to 90.1% for the heterogeneous chip. The runtime of the MILP problem is tractable, it takes only a few minutes for CPLEX to solve these two chips.

As stated before, the manual design has limited search space w.r.t the number of converters, as compared to the two heuristic approaches. For a comparison that is more favorable to the limited search space of manual design, and to explore the quality of our approach under stringent constraints, we perform another set of experiments by setting the same upperbound for the available number of converters for these three approaches.

The results are presented in Table 6.5. Column 3 shows the upper bound for number of converters. From the table we can see that compared to manual design, on average,

Table 6.5: Comparison Of Optimization Efficiency, with same limitation on number of converters

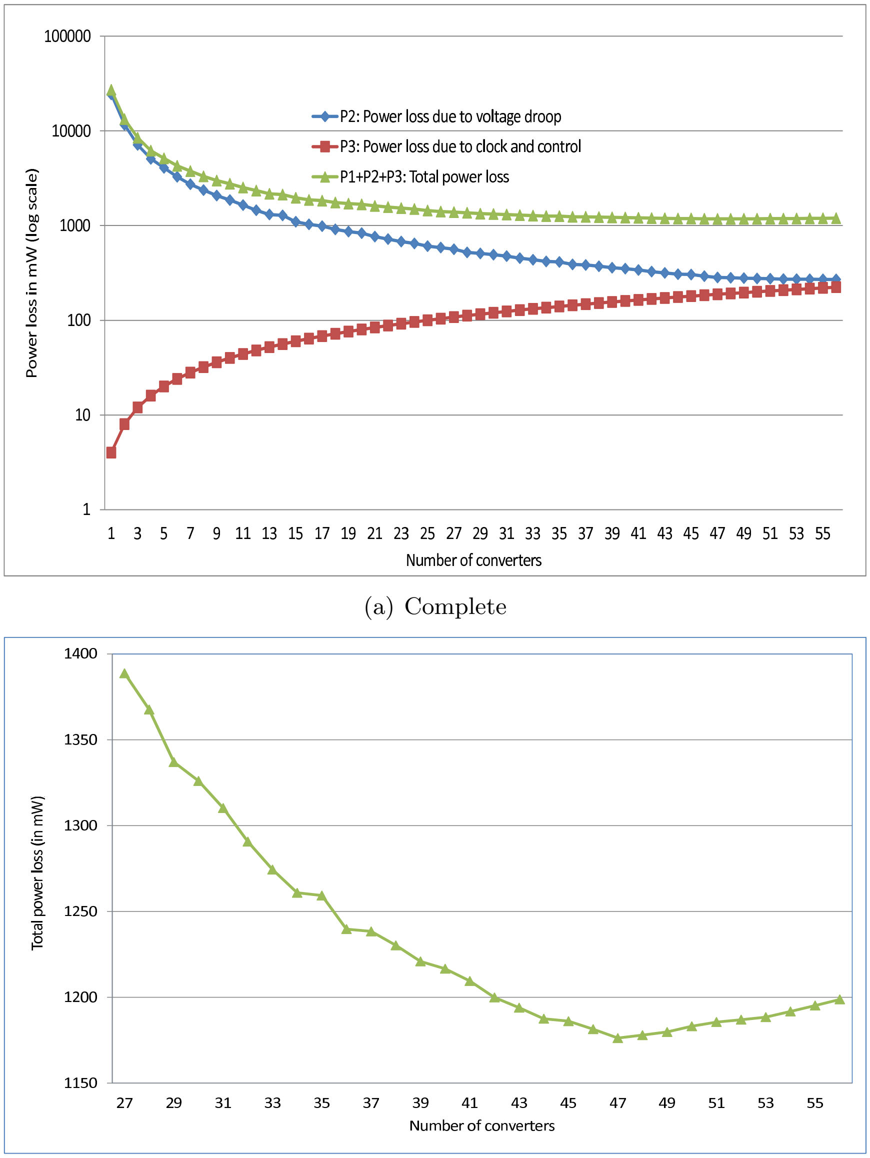
Chip	m	Max. #cvt	n	Manual						Greedy							Heuristic-MILP						
				#cvt	P1	P2	P3	Total	η	#cvt	P1	P2	P3	Total	η	CPU	#cvt	P1	P2	P3	Total	η	CPU
Homo16	56	16	208	16	806	1235	32	2106	81	16	773	1024	64	1861	82.5	2.9	16	779	991	64	1834	82.8	360.4
Hete32	8	76	203	8	160	311	32	503	86	8	158	240	32	430	87.8	1.7	8	157	200	32	389	88.8	374.4
Average						1	1					0.80		0.87					0.72		0.82		

Greedy and *Heuristic-MILP* can still improve the results respectively by 13% and 18% in terms of the total power loss. This is because with the same number of converters, the heuristic approaches can search different combinations of the converters. Even for the homogeneous chip, there is still room for improvement because of the unevenly distribution of current within each core and the asymmetry in the power pads shared by different power domains in a single chip.

Figure 6.8(a) shows how the power losses P_2, P_3 and the total power loss $P_1 + P_2 + P_3$ change with various number of converters for the homogeneous chip by applying the heuristic approach *Heuristic-MILP*. We can see that as we increase the number of converters from 1 (all the cores connected to a converter) to 30, the power loss P_2 due to voltage droop decreases quickly, with a reduction of more than 20X. This implies that the distributed design of the converters can effectively reduce the IR drop seen by the cores, and therefore, improve the efficiency of the power delivery system. The reduction in total power loss starts to slow down as we further increase the converter number, and the overhead from the control circuit and clock network begins to dominate the overall power loss. Similar results can be observed for the heterogeneous chip as shown in Figure 6.9(a).

Figure 6.8(a) shows high power loss (more than 10W) when only a few converters are used. This is because we generated the results with the same wiring resources for different number of converters. The loss number can be reduced by using more interconnect resources through narrowing the pitch of the power grid, but that can cause very high congestion.

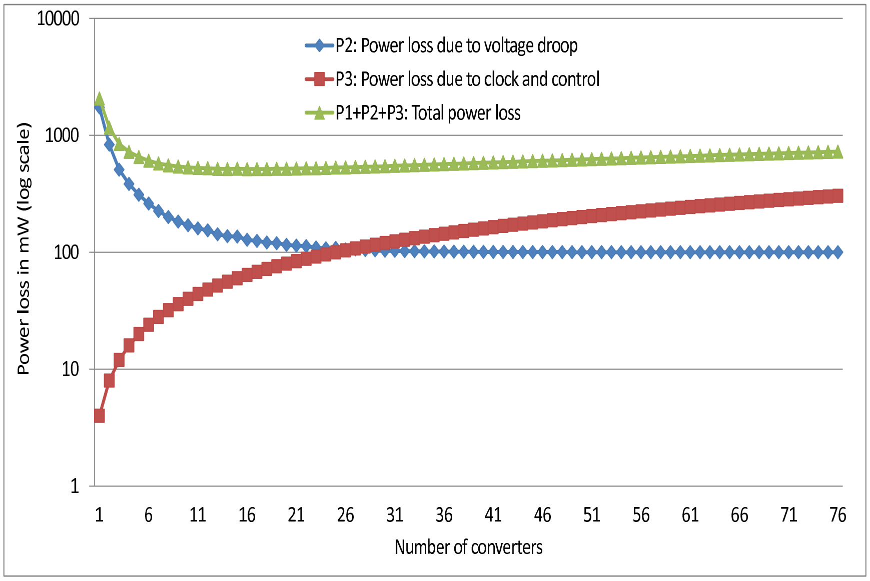
For the homogeneous chip, the lowest total power loss is achieved with 47 converters as shown in Figure 6.8(b), and the layout is shown in Figure 6.7(left). Note that although there is no large difference in the total power loss between the cases using 47 and 56 converters, more routing resource is needed for the clock network when more converters



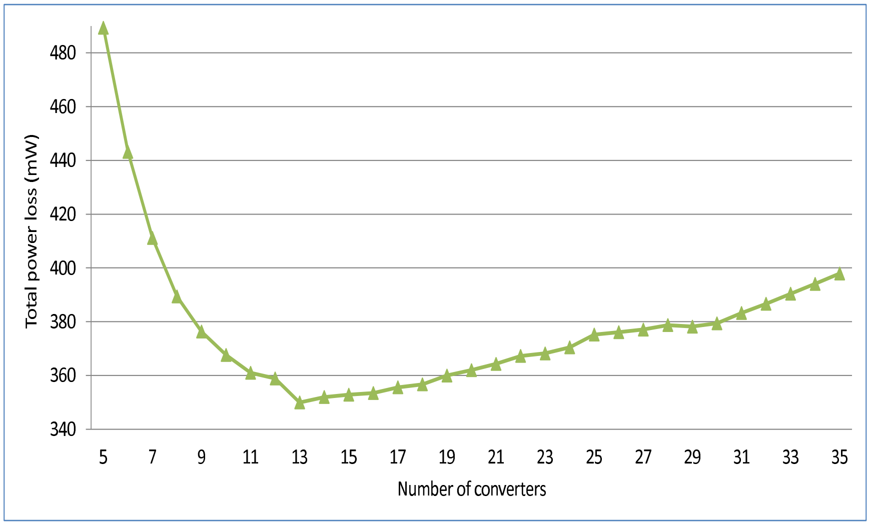
(a) Complete

(b) Part

Figure 6.8: Power loss vs. # converters for homogeneous chip. The left figure shows the complete graph for P_1, P_2 and the total power loss. The right figure shows part of the total power loss as the number of converters changes from 27 to 56.



(a) Complete



(b) Part

Figure 6.9: Power loss vs. # converters for heterogeneous chip. The left figure shows the complete graph for P_1, P_2 and the total power loss. The right figure shows part of the total power loss as the number of converters changes from 5 to 35.

are used, which is not captured by power loss objective function. It is certainly possible to use an enhanced objective that captures this factor, or to determine a reasonable tradeoff by examining the curve. For the heterogeneous chip, the lowest total power loss of is achieved with 13 converters shown in Figure 6.9(b), and the layout is shown in Figure 6.7(right).

Table 6.6: Heuristic-MILP vs. Heuristic-iterative

Chip	Heuristic-MILP						Heuristic-iterative					
	#cvts	P1	P2	P3	Total	CPU	#cvts	P1	P2	P3	Total	CPU
Homo16	47	704.9	283.8	188	1176.7	370.1	47	703.6	283.7	188	1175.3	374.9
Hetero32	13	156.7	141.9	52	350.6	362.7	13	156.1	141.7	52	349.8	364.9

In Section 6.6, we had proposed heuristic approaches to break the MINLP problem (described in Section 6.5) into two independent sub-problems. In fact, we have another formulation (details not shown due to space limitations) that solves MINLP problem approximately in an iterative way: We start with the initial guess to the MINLP problem provided by the *Heuristic-MILP* and closed-form solution presented in Section 6.6.3. And we set the integer variables z_is to be the values from the initial guess (i.e., fixing the number and location of the converters).

The iterative process, called *Heuristic-iterative*, consists of two steps:

(1) For fixed z_is, the MINLP problem in Section 6.5 becomes a NLP, that is solved by CPLEX through sequential linear programming.

(2) We update the number and location of the converters by solving a MILP problem by fixing some variables based on the NLP solution.

The key difference between *Heuristic-MILP* and *Heuristic-iterative* is that we allow the converters to have different voltage ripple ΔV_is in *Heuristic-iterative.* Table 6.6 presents the results of comparison between *Heuristic-MILP* and *Heuristic-iterative.* We observe that *Heuristic-iterative* can only improves the initial guess provided by *Heuristic-MILP* by a small amount. This implies that our assumption about identical voltage ripple made in Section 6.6 is acceptable in terms of the solution quality.

6.8 Conclusion

In this chapter, we study the efficiency of the power delivery system using SC converters at the system level. This work develops models for the efficiency of such a system as a function of size and layout of the SC converters, and the problem is formulated as a mixed integer non-linear program optimization. We then propose heuristic approaches to optimize the size and layout of the SC converter to minimize power loss. The efficiency of these techniques is demonstrated on both homogenous and heterogenous multicore chips.

Chapter 7

Conclusion

This thesis presents CAD solutions to the design and optimization of on-chip communication network and power delivery network for 3D and multicore chips.

In Chapter 2, we have proposed an efficient algorithm to synthesize application-specific 3D NoC architectures for SoCs. Our algorithm utilizes a stochastic approach called simulated allocation (SAL) to reduce the dependency of NoC design results on flow ordering. We also use accurate delay model for routers in NoCs which consider the queueing delay and network contention. Finally, our algorithm performs the floorplanning of cores/routers and NoC topology synthesis in an integrated iterative loop, attempting to find the optimal solution for the problem of application-specific NoC design. Experimental results on a set of benchmarks show that our algorithm can produce greatly improved solutions compared to the baseline algorithm reflecting prior work. We also study the impact of various factors on the network performance in 3D NoCs, including the TSV count and the number of 3D tiers.

For CMPs, we propose flexible-pipeline routers in Chapter 3 that are capable of re-balancing the pipeline stages upon voltage and frequency scaling, while operating the cores at the original frequency. The proposed routers are able to improve energy-efficiency of the system by exploiting the fact that certain workloads are latency-sensitive, but are not throughput-intensive. The proposed technique can work in tandem with (and are largely orthogonal to) other techniques that are intended to reduce router power as well as various congestion management strategies. The hardware complexity for supporting pipeline rebalancing is minimal.

Chapter 4 presents an efficient decap allocation algorithm to optimize 3D power supply network using both MIM and CMOS decaps. Our algorithm uses 3D congestion analysis and a linear congestion model, as well as linearized noise models based on adjoint sensitivity analysis, to guide the decap allocation among CMOS and MIM decaps. Experimental results show that power grid noise can be more effectively optimized using both MIM and CMOS decaps, with lower leakage power and low routing congestion costs.

In the last part of this thesis, we explore the design and optimization of on-chip SC DC-DC converters in multicore processors. Simulation results based on realistic multicore current profiles show that distributed SC converters can reduce the IR drop largely compared to the lumped design, with improved supply voltage. We also present the idea of using SC converters for multi-domain power supply. We then continues to study the efficiency of the power delivery system using SC converters at the system level. We develops models for the efficiency of such a system as a function of size and layout of the SC converters, and the problem is formulated as a mixed integer non-linear program optimization. We then propose heuristic approaches to optimize the size and layout of the SC converter to minimize power loss. The efficiency of these techniques is demonstrated on both homogenous and heterogenous multicore chips.

References

[1] Y. K. Ramadass. *Energy Processing Circuits for Low-Power Applications.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2009.

[2] D. Roberts, W. Johnstone, H. Sanchez, O. Mandhana, D. Spilo, J. Hayden, E. Travis, B. Melnick, M. Celik, B. W. Min, J. Edgerton, M. Raymond, E. Luckowski, C. Happ, A. Martinez, B. Wilson, Pak Leung, T. Garnett, D. Goedeke, T. Remmel, K. Ramakrishna, and B. E. White. Application of on-chip MIM decoupling capacitor for 90nm SOI microprocessor. In *Proceedings of the IEEE International Electron Devices Meeting*, pages 72–75, 2005.

[3] R. J. Riedlinger, R. Bhatia, L. Biro, B. Bowhill, E. Fetzer, P. Gronowski, and T. Grutkowski. A 32nm 3.1 billion transistor 12-wide-issue Itanium® processor for mission-critical servers. In *Proceedings of the IEEE International Solid-State Circuits Conference*, 2011.

[4] International technology roadmap for semiconductors (2010 update). Available at `http://www.itrs.net/Links/2010ITRS/Home2010.htm`.

[5] P. Garrou, C. Bower, and P. Ramm. *Handbook of 3D Integration: Technology and Applications of 3D Integrated Circuits.* New York: Wiley-VCN, May 2008.

[6] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote. Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 28(1):3–21, 2009.

[7] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits*, 43(1):29–41, Jan. 2008.

[8] M. Pióro and P. Gajowniczek. Solving multi commodity integral flow problems by simulated allocation. *Telecommunication Systems*, 7(1-3):17–28, 1997.

[9] S. S. Sapatnekar. CAD for 3D circuits: Solutions and challenges. In *Proceedings of the IEEE International VLSI/ULSI Multilevel Interconnection Conference*, pages 245–251, 2007.

[10] 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. *Proceedings of IEEE*, 89(5), 2001.

[11] L. Xue, C. C. Liu, H.-S. Kim, S. K. Kim, and S. Tiwari. Three-dimensional integration: technology, use, and issues for mixed-signal applications. *IEEE Transactions on Electron Devices*, 50(3):601–609, Mar. 2003.

[12] W. R. Davis et al. Demystifying 3D ICs: The pros and cons of going vertical. *IEEE Design & Test of Computers*, 22(6):498–510, Nov.-Dec. 2005.

[13] S. K. Lim. Physical design for 3D system on package. *IEEE Design & Test of Computers*, 22(6):532–539, Nov.-Dec. 2005.

[14] L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35(1):70–78, 2002.

[15] W. J. Dally and B. Towles. Route packets, not wires: On-chip inteconnection networks. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 684–689, 2001.

[16] S. Murali, D. Atienza, L. Benini, and G. De Micheli. A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 845–848, 2006.

[17] C. Addo-Quaye. Thermal-aware mapping and placement for 3-D NoC designs. In *IEEE International SOC Conference*, pages 25–28, 2005.

[18] V. F. Pavlidis and E. G. Friedman. 3-D topologies for networks-on-chip. *IEEE Transactions on Very Large Scale Integration Systems*, 15(10):1081–1090, 2007.

[19] B. Feero and P. P. Pande. Performance evaluation for three-dimensional networks-on-chip. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 305–310, 2007.

[20] 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 *Proceedings of the ACM International Symposium on Computer Architecture*, pages 138–149, 2007.

[21] H. Matsutani, M. Koibuchi, and H. Amano. Tightly-coupled multi-layer topologies for 3-D NoCs. In *Proceedings of the International Conference on Parallel Processing*, pages 75–75, 2007.

[22] 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 *Proceedings of the International Symposium on Computer Architecture*, pages 130–141, 2006.

[23] P. Morrowa1, B. Blacka, M. J. Kobrinskya, S. Muthukumara, D. Nelsona, C.-M. Parka, and C. Webb. Design and fabrication of 3D microprocessors. In *Materials Research Society Symposium*, 2007.

[24] S. Yan and B. Lin. Design of application-specific 3D networks-on-chip architectures. In *Proceedings of the International Conference on Computer Design*, pages 142–149, 2008.

[25] S. Murali, C. Seiculescu, L. Benini, and G. D. Micheli. Synthesis of networks on chips for 3D systems on chips. In *Proceedings of the Asia-South Pacific Design Automation Conference*, pages 242–247, 2009.

[26] T. Ahonen, H. Bin, and J. Nurmi. Topology optimization for application-specific networks-on-chip. In *Proceedings of the International Workshop on System Level Interconnect Prediction*, pages 53–60, 2004.

[27] K. Srinivasan, K. S. Chatha, and G. Konjevod. An automated technique for topology and route generation of application specific on-chip interconnection networks. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 231–237, 2005.

[28] A. Hansson, K. Goossens, and A. Rădulescu. A unified approach to constrained mapping and routing on network-on-chip architectures. In *Proceedings of the International Conference on Hardware-Software Codesign and System Synthesis*, pages 75–80, 2005.

[29] S. Yan and B. Lin. Application-specific network-on-chip architecture synthesis based on set partitions and Steiner trees. In *Proceedings of the Asia-South Pacific Design Automation Conference*, pages 277–282, 2008.

[30] C. Seiculescu, S. Murali, L. Benini, and G. D. Micheli. SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chip. In *Proceedings of Design and Test in Europe*, pages 9–14, 2009.

[31] Y. Hu, H. Chen, Y. Zhu, A. A. Chien, and C.-K. Cheng. Physical synthesis of energy-efficient networks-on-chip through topology exploration and wire style optimizations. In *Proceedings of the International Conference on Computer Design*, pages 111–118, 2005.

[32] Y. Hu, Y. Zhu, H. Chen, R. Graham, and C.-K. Cheng. Communication latency aware low power NoC synthesis. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 574–579, 2006.

[33] D. S. Mark, M. Karpovsky, and L. Zakrevski. Application of network calculus to general topologies using turn-prohibition. *IEEE/ACM Transactions on Networking*, 11(3):411–421, 2003.

[34] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[35] U. Y. Ogras and R. Marculescu. Analytical router modeling for networks-on-chip performance analysis. In *Proceedings of Design and Test in Europe*, pages 1096 – 1101, 2007.

[36] J. D. C. Little. A proof for the queuing formula: L= λW. *Operations Research*, 9(3):383–387, 1961.

[37] W.-L. Hung, G. M. Link, Yuan Xie, N. Vijaykrishnan, and M. J. Irwin. Interconnect and thermal-aware floorplanning for 3D microprocessors. In *Proceedings of the International Symposium on Quality Electronic Design*, pages 98–104, 2006.

[38] H. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *Proceedings of the annual IEEE/ACM International Symposium on Microarchitecture*, pages 294–305, 2002.

[39] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. Orion 2.0: a fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 423–428, 2009.

[40] I. Loi, F. Angiolini, and L. Benini. Supporting vertical links for 3D networks-on-chip: toward an automated design and analysis flow. In *Proceedings of Nano-Net*, pages 1–5, 2007.

[41] D. Greenfield, A. Banerjee, J.-G. Lee, and S. Moore. Implications of Rent's Rule for NoC design and its fault-tolerance. In *ACM/IEEE International Symposium on Networks-on-Chip*, pages 283–294, 2007.

[42] H. Wang, L.-S Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of the annual IEEE/ACM International Symposium on Microarchitecture*, page 105, 2003.

[43] R. Kumar, D. M. Tullsen, N. P. Jouppi, and P. Ranganathan. Heterogeneous Chip Multiprocessors. *Computer*, 38:32–38, 2005.

[44] F. A. Bower, D. J. Sorin, and L. P. Cox. The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling. *IEEE Micro*, 28(3):17–25, May. 2008.

[45] U. Y. Ogras and R. Marculescu. It's a small world after all: NoC Performance Optimization via Long-range Link Insertion. *IEEE Transactions on VLSI Systems*, 14(7):693–706, 2006.

[46] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha. Express Virtual Channels: Towards the Ideal Interconnection Fabric. In *Proceedings of the ACM International Symposium on Computer Architecture*, pages 150–161, 2007.

[47] M. F. Chang, J. Cong, A. Kaplan, M. Naik, G. Reinman, E. Socher, and S.-W. Tam. CMP Network-on-chip Overlaid with Multi-band RF-interconnect. pages 191–202, 2008.

[48] L.-S. Peh and W. J. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. pages 255–266, 2001.

[49] R. Mullins, A. West, and S. Moore. The Design and Implementation of a Low-Latency On-Chip Network. In *Proceedings of the Asia-South Pacific Design Automation Conference*, pages 164–169, 2006.

[50] A. Kumar, P. Kundu, A. P. Singh, L.-S. Peh, and N. K. Jha. A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator. In *Proceedings of the IEEE International Conference on Computer Design*, pages 63–70, 2007.

[51] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks. pages 91–102, 2003.

[52] E. J. Kim, K. H. Yum, G. M. Link, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, M. Yousif, and C. R. Das. Energy Optimization Techniques in Cluster Interconnects. In *Proceedings of the ACM International Symposium on Low Power Electronics and Design*, pages 459–464, 2003.

[53] S. E. Lee and N. Bagherzadeh. A Variable Frequency Link for a Power-aware Network-on-chip (NoC). *Integration, the VLSI Journal*, 42(4):479–485, 2009.

[54] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das. A Case for Dynamic Frequency Tuning in On-chip Networks. pages 292–303, 2009.

[55] Y. Hirata, H. Matsutani, M. Koibuchi, and H. Amano. A Variable-pipeline On-chip Router Optimized to Traffic Pattern. pages 57–62, 2010.

[56] N. Agarwal, L.-S. Peh, and N. Jha. Garnet: A Detailed Interconnection Network Model inside a Full-system Simulation Framework. Technical report, Princeton University, 2008.

[57] S. Sapatnekar. *Timing.* Kluwer Academic Publishers, Boston, MA, 2004.

[58] T Sakurai and A R Newton. Alpha-power Law MOSFET Model and Its Applications to CMOS Inverter Delay and Other Formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–594, 1990.

[59] B. M. Beckmann M. R. Marty M. Xu A. R. Alameldeen K. E. Moore M. D. Hill D. A. Wood M. M. K. Martin, D. J. Sorin. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. *SIGARCH Computer Architecture News*, 33:92–99, 2005.

[60] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, Feb. 2002.

[61] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of the ACM International Symposium on Computer Architecture*, pages 83–94, 2000.

[62] SPEC OMP2001. Available at `http://www.spec.org/omp/`.

[63] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, J. Pisharath, G. Memik, and A. Choudhary. MineBench: A Benchmark Suite for Data Mining Workloads. pages 182–188, 2006.

[64] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. pages 72–81, 2008.

[65] H. Su, S. S. Sapatnekar, and S. R. Nassif. An algorithm for optimal decoupling capacitor sizing and placement for standard cell layouts. In *Proceedings of the International Symposium on Physical Design*, pages 68–73, 2002.

[66] H. H. Chen and D. D. Ling. Power supply noise analysis methodology for deep-submicron VLSI chip design. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 638–643, 1997.

[67] E. Wong, J. Minz, and S. K. Lim. Decoupling capacitor planning and sizing for noise and leakage reduction. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 395–400, 2006.

[68] J. R. Minz, S. K. Lim, and C.-K. Koh. 3D module placement for congestion and power noise reduction. In *Proceedings of the Great Lakes Symposium on VLSI*, pages 458–461, 2005.

[69] G. Huang, M. Bakir, A. Naeemi, H. Chen, and J. D. Meindl. Power delivery for 3D chip stacks: Physical modeling and design implication. In *Proceedings of the IEEE Electrical Performance of Electronic Packaging Meeting*, pages 205–208, 2007.

[70] Y. L. Tu, H. L. Lin, L. L. Chao, D. Wu, C. S. Tsai, C. Wang, C. F. Huang, C. H. Lin, and J. Sun. Characterization and comparison of high-k metal-insulator-metal (MiM) capacitors in 0.13μm Cu BEOL for mixed-mode and RF applications. In *Proceedings of the IEEE International Symposium on VLSI Circuits*, pages 79–80, 2003.

[71] H. Sanchez, B. Johnstone, D. Roberts, O. Mandhana, B. Melnick, M.Celik, M. Baker, J. Hayden, Byoung Min, J. Edgerton, and B. White. Increasing microprocessor speed by massive application of on-die high-k MIM decoupling capacitors. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 2190–2199, 2006.

[72] P. Zhou, Y. Ma, Z. Li, R. P. Dick, L. Shang, H. Zhou, X. Hong, and Q. Zhou. 3D-STAF: Scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 590–597, 2007.

[73] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng. Estimating routing congestion using probabilistic analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(1):32–41, January 2002.

[74] B. Goplen and S. S. Sapatnekar. Placement of 3D ICs with thermal and interlayer via considerations. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 626–631, 2007.

[75] Predictive technology model. Device Group at Arizona State University, Available at `http://www.eas.asu.edu/~ptm`.

[76] G. Patounakis, Y. W. Li, and K. L. Shepard. A fully integrated on-chip DC-DC conversion and power management system. *IEEE Journal of Solid-State Circuits*, 39(3):443–451, Mar. 2004.

[77] Z. Zeng, X. Ye, Z. Feng, and P. Li. Tradeoff analysis and optimization of power delivery networks with on-chip voltage regulation. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 831–836, Jun. 2010.

[78] Y. K. Ramadass and A. P. Chandrakasan. Voltage scalable switched capacitor DC-DC converter for ultra-low-power on-chip applications. In *IEEE Power Electronics Specialists Conference*, pages 2353–2359, Jun. 2007.

[79] Y. Ramadass, A. Fayed, B. Haroun, and A. Chandrakasan. A 0.16mm^2 completely on-chip switched-capacitor DC-DC converter using digital capacitance modulation for LDO replacement in 45nm CMOS. In *IEEE International Solid-State Circuits Conference*, pages 208–209, Feb. 2010.

[80] H.-P. Le, M. Seeman, S. R. Sanders, V. Sathe, S. Naffziger, and E. Alon. A 32nm fully-integrated reconfigurable switched-capacitor DC-DC converter delivering 0.55 W/mm^2 at 81% efficiency. In *IEEE International Solid-State Circuits Conference*, pages 210–211, Feb. 2010.

[81] L. Chang, R. K. Montoye, B. L. Ji, A. J. Weger, K. G. Stawiasz, and R. H. Dennard. A fully-integrated switched-capacitor 2:1 voltage converter with regulation capability and 90% efficiency at 2.3A/mm^2. In *IEEE Symposium on VLSI Circuits*, pages 55–56, Jun. 2010.

[82] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. Mackay, M. Reif, Liewei Bao, J. Brown, M. Mattina, Chyi-Chang Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook. Tile64 - processor: A 64-core SoC with mesh interconnect. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 88–598, 2008.

[83] H.-P. Le, S. R. Sanders, and E. Alon. Design techniques for fully integrated switched-capacitor DC-DC converters. *IEEE Journal of Solid-State Circuits*, 46(9):2120–2131, Sept. 2011.

[84] S. S. Kudva and R. Harjani. Fully-integrated on-chip DC-DC converter with a 450x output range. *IEEE Journal of Solid-State Circuits*, 46(8):1940–1951, Aug. 2011.

[85] D. Duarte, N. Vijaykrishnan, and M. J. Irwin. A complete phase-locked loop power consumption model. In *Proceedings of the Design, Automation & Test in Europe*, page 1108, 2002.

[86] P. Zhou, D. Jiao, C. H. Kim, and S. S. Sapatnekar. Exploration of on-chip switched-capacitor DC-DC converter for multicore processors using a distributed power delivery network. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 1–4, 2011.

[87] M. Zhao, R. V. Panda, S. S. Sapatnekar, T. Edwards, R. Chaudhry, and D. Blaauw. Hierarchical analysis of power distribution networks. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 150–155, 2000.

[88] M. R. Bussieck and A. Pruessner. Mixed-integer nonlinear programming. *SIAG/OPT Newsletter: Views & News*, 2003.

[89] IBM ILOG CPLEX Optimization Studio v.12. Available at `http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/`.

[90] ARM Cortex processors. Available at `http://arm.com/products/processors/index.php`.

[91] ARM Holdings plc. big.LITTLE Processing. Available at `http://www.arm.com/products/processors/technologies/biglittleprocessing.php`.