2016

Increasing Off-Chip Bandwidth and Mitigating Dark Silicon via Switchable Pins

Shaoming Chen
Louisiana State University and Agricultural and Mechanical College

INCREASING OFF-CHIP BANDWIDTH AND MITIGATING DARK SILICON VIA SWITCHABLE PINS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

School of Electrical Engineering and Computer Science

by
Shaoming Chen
B.E., Huazhong University of Science and Technology, Wuhan, China 2008
M.E., Huazhong University of Science and Technology, Wuhan, China 2011
August 2016

ACKNOWLEDGEMENTS

I would like to dedicate the dissertation to my parents and my friends for their continuous support and encouragement throughout my entire life.

This dissertation is completed with the valuable help and support from a lot of people including my advisor, Dr. Lu Peng. He thoughtfully guided me to pick up the emerging topic and offered valuable advises for my study. Dr. Ashok Srivastava gave appreciated help of circuit design and encouraged me to response bitter feedbacks from reviewers. Dr. Bin Li from Department of Experimental Statistics provided insightful statistical methods to help me to analyze experimental data. Dr. David Koppelman and Dr. Rudy Hirschheim as my committee members take their valuable time to supervise my dissertation and attend my defense. I sincerely appreciate the professors' supports as the foundation of the work. I also would like to thanks my co-workers in my lab. Dr. Zhang provided insight thoughts that helped me to develop the dissertation. Yue devoted enormous effort on circuit design especially on power delivery network. Zhou elaborated the circuit design and helped me to improve the circuit design. Sam proofread the work and gave me wonderful feedbacks.

I am thankful to the Department of Electrical and Computer Engineering for providing assistantship throughout my study.

Finally, I would like thank all the friends I met at LSU for making my life here wonderful and memorable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Off-chip memory bandwidth has been considered as one of the major limiting factors to processor performance, especially for multi-cores and many-cores. Conventional processor design allocates a large portion of off-chip pins to deliver power, leaving a small number of pins for processor signal communication. We observed that the processor requires much less power than that can be supplied during memory intensive stages in some cases. In this work, we propose a dynamic pin switch technique to alleviate the bandwidth limitation issue. The technique is introduced to dynamically exploit the surplus pins for power delivery in the memory intensive phases and uses them to provide extra bandwidth for the program executions, thus significantly boosting the performance. We also explore its performance benefit in the era of Phase-change memory (PCM) and prove that the technique can be applied beyond DRAM-based memory systems.

On the other hand, the end of Dennard Scaling has led to a large amount of inactive or significantly under-clocked transistors on modern chip multi-processors in order to comply with the power budget and prevent the processors from overheating. This so-called "dark silicon" is one of the most critical constraints that will hinder the scaling with Moore's Law in the future. While advanced cooling techniques, such as liquid cooling, can effectively decrease the chip temperature and alleviate the power constraints; the peak performance, determined by the maximum number of transistors which are allowed to switch simultaneously, is still confined by the amount of power pins on the chip package. In this paper, we propose a novel mechanism to power up the dark silicon by dynamically switching a portion of I/O pins to power pins when off-chip communications are less frequent. By enabling extra cores or increasing processor frequency, the proposed strategy can significantly boost performance compared with traditional designs.

Using the switchable pins can increase inter-socket bandwidth as one of performance bottlenecks. Multi-socket computer systems are popular in workstations and servers. However, they suffer from the relatively low bandwidth of inter-socket communication especially for massive parallel workloads that generates many inter-socket requests for synchronizations and remote memory accesses. The inter-socket traffic poses a huge pressure on the underlying networks fully connecting all processors with the limited bandwidth that is confined by pin resources. Given the constraint, we propose to dynamically increase the inter-socket bandwidth, trading off with lower off-chip memory bandwidth when the systems have heavy inter-socket communication but few off-chip memory accesses. The design increases the physical bandwidth of inter-socket communication via switching the function of pins from off-chip memory accesses to inter-socket communication.

CHAPTER 1. INTRODUCTION

1.1. MOTIVATIONS

As memory-intensive applications such as web servers, database software, and tools for data analysis prevail, the focus of computer architects shifts from Instruction Level Parallelism (ILP) to Memory Level Parallelism (MLP). The term “Memory Wall” was coined to describe the disparity between the rate of core performance improvement and the relatively stagnant rate of off-chip memory bandwidth increase. Additional cores, when integrated on the same die, and supplemental applications serve to widen this gap, since each individual core may generate substantial memory requests that need to be queued and served by the memory subsystem. Obviously, the capability of the off-chip memory system largely determines the per-core or even the overall performance of the entire system. In scenarios where the off-chip memory is insufficiently fast to handle all memory transactions in a timely manner, the system performance is highly likely to be bottlenecked by the slow memory accesses. An intuitive solution to this problem is to increase the off-chip memory bandwidth by enabling more memory channels. Figure 1-1 illustrates the variation of normalized throughput with the number of memory channels increased from 1 to 4 when 4 lbm programs are running on an X86 platform. As can be seen from the figure, enabling more memory channels significantly increases the off-chip bandwidth, which in turn translates to an impressive boost of the system performance. Furthermore, compared to compute-intensive stages, processors consume much less power during memory-intensive phases when cores wait for data to be fetched from main memory.

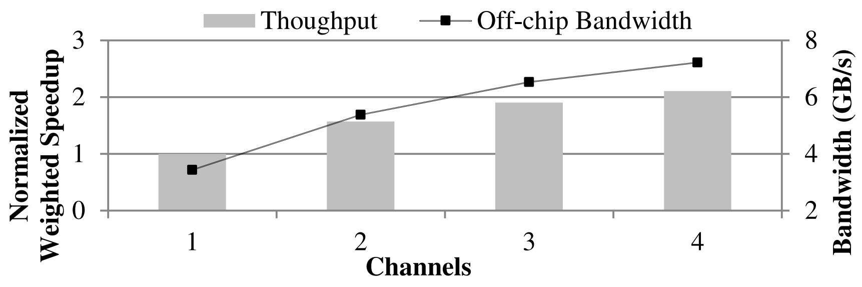


Figure 1-1. Normalized weighted speedup and off-chip bandwidth of 4 lbm co-running on a processor with 1,2,3,4 memory channels

Motivated by this observation, we propose an innovative technique to mitigate the shortage of off-chip bandwidth during the memory-intensive phases of program executions, in order to enhance the overall performance. Our scheme is built on top of a novel switchable pin design and accurate identifications of memory-intensive phases. Pins can be dynamically altered for power delivery or signal transmission via accessory circuits. These circuits enable pins to deliver quality power or signal with relatively low area overhead. On the other hand, we identify the memory-intensive phases by observing the key performance metrics at runtime. Extra off-chip bandwidth is demanding in phases with high memory intensity. Therefore, by switching the pins and providing additional bandwidth for off-chip memory transactions, the performance of memory-intensive stages can be boosted, thus impressively accelerating the overall execution.

On the other hand, "dark silicon" can be mitigated via the switchable pins. In the current industry, there are two commonly accepted reasons for power constraints that cause dark silicon: thermal constraints and power delivery [41]. The slow improvement of per-transistor switch energy along with the fast growing transistor density has led to a considerable rise in the power consumption per unit area (i.e., power density). Provided that inexpensive cooling techniques such as air cooling are still the mainstream solution to heat dissipation for desktop and mobile platforms, such increasing of the power density tends to generate substantial heat that outstrips the chip's heat spreading capability. In this situation, the maximum power consumption of the chip cannot go beyond a threshold in order to maintain a safe working temperature for the entire processor. This power limit is usually referred to as the thermal design power (TDP). Some high-end processors with a higher TDP use backplate liquid cooling [33] to avoid thermal issues.

The underlying power delivery system, on the other hand, constrains the amount or the frequency of simultaneously active transistors as it determines the maximum power that is able to be provided to the chip irrespective of the thermal concern. To alleviate this constraint, we consider increasing the power envelope with minimum circuit change to the existing computer systems, in order to enable more transistors or raise the operating frequency in the power-hungry phases during program execution. Figure 1-2 plots a snapshot of the execution of 8 copies of DEALII from SPEC2006 on an 8-core processor, visualizing a representative scenario that motivates our work. The off-chip memory traffic and processor power consumption both vary in different execution phases. More interestingly, the two traces generally show an opposite trend during the execution; when the memory traffic is relatively light, the total power consumption is quite considerable

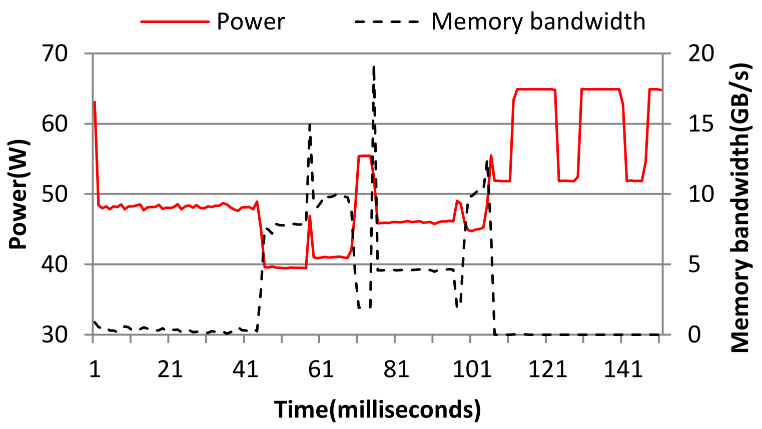


Figure 1-2. Power and memory bandwidth (8 copies of DEALII from SPEC2006)

(e.g., time interval 106 – 151ms). On the other hand, a duration of memory-intensive execution will correspond to a low-power period. The underlying reason for this phenomenon is that frequent misses in the last-level cache and the resultant off-chip memory accesses will largely slow down the overall execution rate, leading to a decrease in the processor's power consumption.

This intuitive observation implies an important opportunity for performance improvement and dark silicon mitigation by appropriately balancing the power delivery and off-chip traffic. To exploit this potential benefit, we propose a novel mechanism to dynamically switch a portion of I/O pins for extra power delivery when off-chip memory accesses are infrequent, thus powering up the dark silicon for performance boost. During a phase when off-chip activities are relatively high, we switch back the pins for signal transmission.

Pin Switching provides a great opportunity for increasing the off-chip bandwidth of CPUs using Phase-change memory (PCM). As DRAM is experiencing difficulties with memory technology scaling, architects are intensively studying potential alternative memory technologies such as PCM. Although PCM exhibits different features from DRAM, Pin Switching is expected to also improve the performance of PCM subsystems. This work investigates the potential benefit of Pin Switching in the era of PCM.

Pin Switching also provides an opportunity for increasing the inter-socket bandwidth as one of performance bottlenecks. Multi-socket systems are widely used to boost the throughput of massive parallel workloads that generate intensive local traffic, between processors and off-chip memory devices such as DRAM, and remote traffic for inter-socket communication. The limited local bandwidth of main memory bounds the performance of parallel workloads, since it serializes the parallel memory requests and offsets the benefit of memory level parallelism, especially considering the ever-increasing data size of workloads and number of cores per die. This problem is addressed by many architects by boosting the system throughput via advanced algorithms for off-chip memory requests [11], increasing the physical memory bandwidth at the cost of lower core frequency [6], or reducing traffic via using a stacked DRAM, which has higher bandwidth than off-chip memory devices and a larger size than a SRAM-based cache [18]. These solutions relieve the performance bottleneck, while remote inter-socket bandwidth emerges as a new performance bottleneck for workloads with intensive inter-socket communication.

Remote bandwidth bounds the performance of workloads that frequently fetch data from the cache of other processors or remotely from main memory. Inadequate remote bandwidth serializes memory requests and limits the benefits of memory level parallelism. The bottleneck of inter-socket communications such as QuickPath Interconnect (QPI) [20] was hidden, as remote main memory access is constrained by off-chip bandwidth, but is now revealed by the volume of requests directly to the DRAM

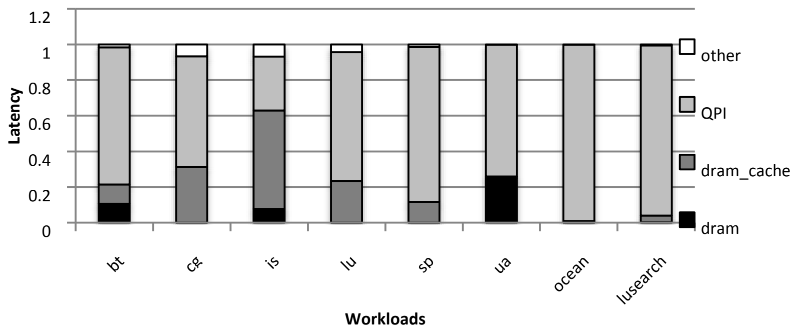


Figure 1-3. The latency breakdown of un-core requests in the simulated system with two sockets

cache that do not use off chip bandwidth. The QPI bandwidth becomes a greater concern than off chip bandwidth when data is more likely to be fetched from stacked DRAM, which has superior bandwidth compared to the remote bandwidth. This bottleneck is shown in Figure 1-3 that breaks down the latencies of un-core requests.

qSwitch, which dynamically allocates off-chip bandwidth between local and remote accesses, is proposed to relieve the bottleneck constraining remote accesses. The total number of pins bound the bandwidth as a scarce resource [24] that power delivery networks and I/O compete for. Additionally, increasing the total number of signal pins is prohibitive since routing traces beneath processors is becoming very difficult. qSwitch dynamically shifts a portion of local off-chip bandwidth for accessing main memory into remote inter-socket communication bandwidth when low local access activities are observed without increasing the total number of signal pins. qSwitch improves the performance of workloads suffering from limited inter-socket bandwidth, based on a vertical design from the circuit to architecture level.

1.2. DISSERTATION ORGANIZATION

The dissertation first presents a pin switch technique to increase off-chip bandwidth based on switchable pins. It demonstrates applying the switchable pins to mitigate dark silicon by boosting core frequency. Additionally, it explores the benefit of the pin switch technique in the era of PCM with multi-threaded workloads. Based on the underlying idea of the pin switch technique, it proposes another pin switch technique to increase inter-socket bandwidth. In general, the main contributions of this work are summarized as follows:

- We propose a switchable pin design which can convert a power pin to a signal pin or the other way around for increasing off-chip bandwidth. Detailed examinations at both the circuit and architectural level are conducted to validate the feasibility of the proposed design. We examine the performance improvement of the design in various memory configurations. A sensitivity study is conducted to compare the benefit of our design with a different number of channels, buses, banks and ranks. We design Dynamic Switching to alleviate the negative side-effects of pin switching by actively identifying memory-intensive phases and only switching when the condition is satisfied. Without prior knowledge of program characteristics, this policy switches the system to prioritize memory bandwidth or core performance according to the identified phase. Our experiments show that significant performance improvement can be achieved for memory-intensive workloads while maintaining the same performance for compute-intensive workloads as the system without Pin Switching.
- We give a circuit implementation for mitigating dark silicon, using minor changes to existing processor and motherboard circuitry. We further design a rigorous statistical model that correlates the historical execution behaviors and off-chip access intensities in upcoming intervals. The established model can be employed by the operating system or equivalent supervisor to guide pin switching at runtime. We conduct a series of simulations to evaluate the performance, energy efficiency, and thermal impact of the proposed design on a chip multi-processor (CMP) in the dim silicon [81].

- We integrate a PCM model into our simulations to evaluate the benefits of Pin Switching in the era of PCM. Pin Switching significantly improves the performance of the PCM memory subsystem in our evaluation. We also show that multi-threaded workloads can benefit from Pin switching as long as they share the performance bottleneck of off-chip bandwidth.
- We identify that the latency of inter-socket communication as the major bottleneck for massive parallel workloads that intensively share data across sockets. We propose qSwitch for improving the performance of the workloads on a multi-socket system in which switching agents turn on/off memory channels, QPI buses, and off-chip bus connections. We evaluate the performance of qSwitch with the selected multi-thread workloads. We also investigate the runtime overhead and signal integrity for qSwitch.

The remainder of the dissertation is organized as follows. We present the design which increase off-chip bandwidth via switchable pins in chapter 2, and the design which mitigate dark silicon in chapter 3. We propose boosting off-chip bandwidth with PCM and improve the performance of multi-threaded programs in chapter 4. Finally, we propose increasing inter-socket bandwidth via switchable pins in chapter 5.

CHAPTER 2. INCREASING OFF-CHIP BANDWIDTH IN MULTI-CORE PROCESSORS WITH SWITCHABLE PINS

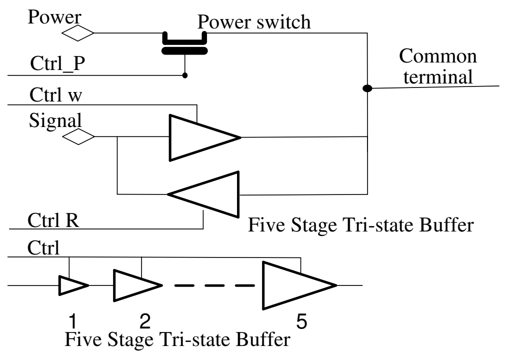
2.1. DESIGN OVERVIEW

Our design aims to boost computer system performance especially for memory-intensive programs. In conventional designs, the performances of these workloads are degraded by a shortage of memory buses which limits off-chip bandwidth. We provide increased memory bandwidth, thereby reducing the average latency of off-chip memory access, at the expense of a lower core frequency. Rather than retaining a fixed number of buses connected to the DRAM (typically one bus per channel), our design dynamically switches buses between signal and power pins (VDD or GND) to reduce the latency for these workloads. This is referred to as multi-bus mode henceforth, as opposed to single-bus mode similar to conventional processor operation. Switchable pins facilitate changing between these two modes as discussed below. This paper focuses on how to fully exploit the benefits of substituting power pins for I/O pins during memory-intensive programs without interfering with compute-intensive programs.

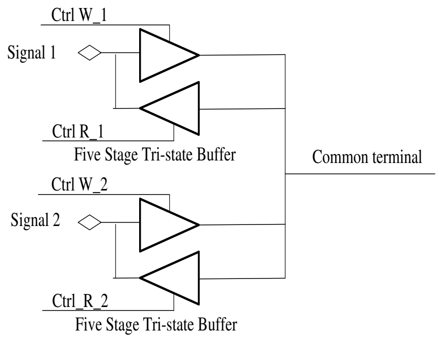
2.1.1 Pin Switch

Figure 2-1 depicts the schematic of two switches and a signal buffer which serve as the basic units for exchanging power pins for signal pins. The signal-to-power switch shown in

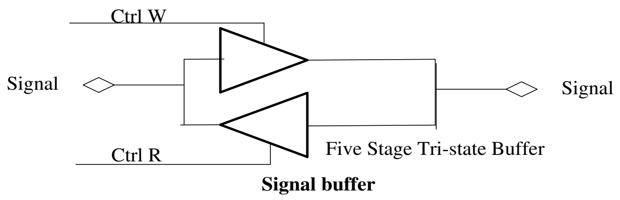
Figure 2-1 (a) is key to alternate a regular pin between the two modes. As illustrated in this figure, we utilize a dedicated power switch [59] which sits on the power delivery path to minimize the corresponding IR drop and power consumption with its



(a) The circuit of a signal-to-power switch



(b) The circuit of a signal switch



(c) The circuit of a signal buffer

Figure 2-1. The circuit of pin switch

ultra-low switch-on resistance, measuring as low as 1.8mΩ. While in the single-bus mode, the power switch is turned on while two 5 stage tri-state buffers on the signal line are off. Otherwise, the power switch is turned off to block noisy interference from the power line, and the tri-state buffers are turned on in one direction according to whether data is read from the memory or written by the memory controller. To compensate for the parasitic capacitances of the power switch, we place the 5 stage tri-state buffers in signal lines to amplify I/O signals. Between each stage, the buffer size is increased by four times to amplify the signal with small delay. In total, the 5 stage tri-state buffer incurs a 0.9ns delay. On the other hand, the die area of the aforementioned power switch is commensurate to that of 3,000 traditional transistors [59]. The number of signal pins for a DRAM bus could slightly vary depending on different processors (e.g. with or without ECC). We pick up 125 power switches per bus which consists of 64 data pins and 61 address and command pins from the pin allocation of an i5-4670 Intel Processor [7]. The total die area consumes 375,000 (3,000 * 125) traditional transistors. Considering a billion-transistor chip, the area overhead for the 3 buses which will be used in our work is less than 0.12% of the total chip area.

The signal switch shown in Figure 2-1 (b) is employed to guarantee that data in the DRAM can be accessed in two modes. The signal switch uses two pairs of 5 stage tri-state buffers to enable memory devices that can be accessed via two buses. The buffers identical to that in the signal-to-power switch can resist noise from a channel when the other channel is selected. On the other hand, the signal buffers shown in Figure 2-1 (c) also have strong peak-drive current and sink capabilities. They are utilized to amplify the signal in order to offset the effect of the parasitic capacitance.

Processors possess specific pin allocations depending on the package, power consumption, and hardware interface (the number of memory channels). For our experiment, we use the pin allocation of an i5-4670 Intel Processor [7] shown in Table 2-1. While this processor includes 4 cores and 2 memory channels, 54.6% of the pins are used for power delivery. Out of the 628 power pins, 125 of these can be replaced with switchable pins for a single bus. To maintain the same ratio of VDD to GND pins, we allocate 30 of the 125 switchable pins as VDD pins and the remaining 95 as GND pins. In our experiment we will allocate at most three additional buses via pin switching because adding more leads to a considerable drop in performance.

2.1.2 Off-Chip Bus Connection

Designing a memory interface which could take the advantage of the switchable pins to dynamically increase off-chip bandwidth is non-trivial. In this section, we propose an off-chip bus connection and instructions to configure the switchable pins for power delivery or for signal transmission.

The two modes of the off-chip bus connection could be described as the multi-bus mode and the single-bus mode, as shown in Figure 2-2. In multi-bus mode, several buses (assuming N) are connected to private DRAM interfaces via the individual buses. On the other hand, single-bus mode can only access DRAM by a single bus. Two signal-to-power switches and a signal switch for each signal wire of N-1 buses are needed. These signal-to-power switches configure the switchable pins for signal transmission where the signal switches connect the bus to DRAM devices in the multi-bus mode, otherwise the switchable pin is configured for power delivery where the DRAM devices are connected to the shared bus.

In order to implement the mechanism, we control the signal-to-power switch detailed in Figure 2-1 (a) and the signal switch detailed in Figure 2-1 (b) to route signal and power in the two modes. The signal to the DRAM interface could be divided into two groups: command signals and data signals. The command signals running in one direction could be routed via the two switches which only need one direction buffer instead of a pair. On the other hand, the data signals (DQ) are bi-directional and the switches shown in Figure 2-2 could receive and send signals in both directions.

For the placements of the switches on the printed circuit board (PCB), one signal-to-power switch for each signal line should be placed close to the processor package in

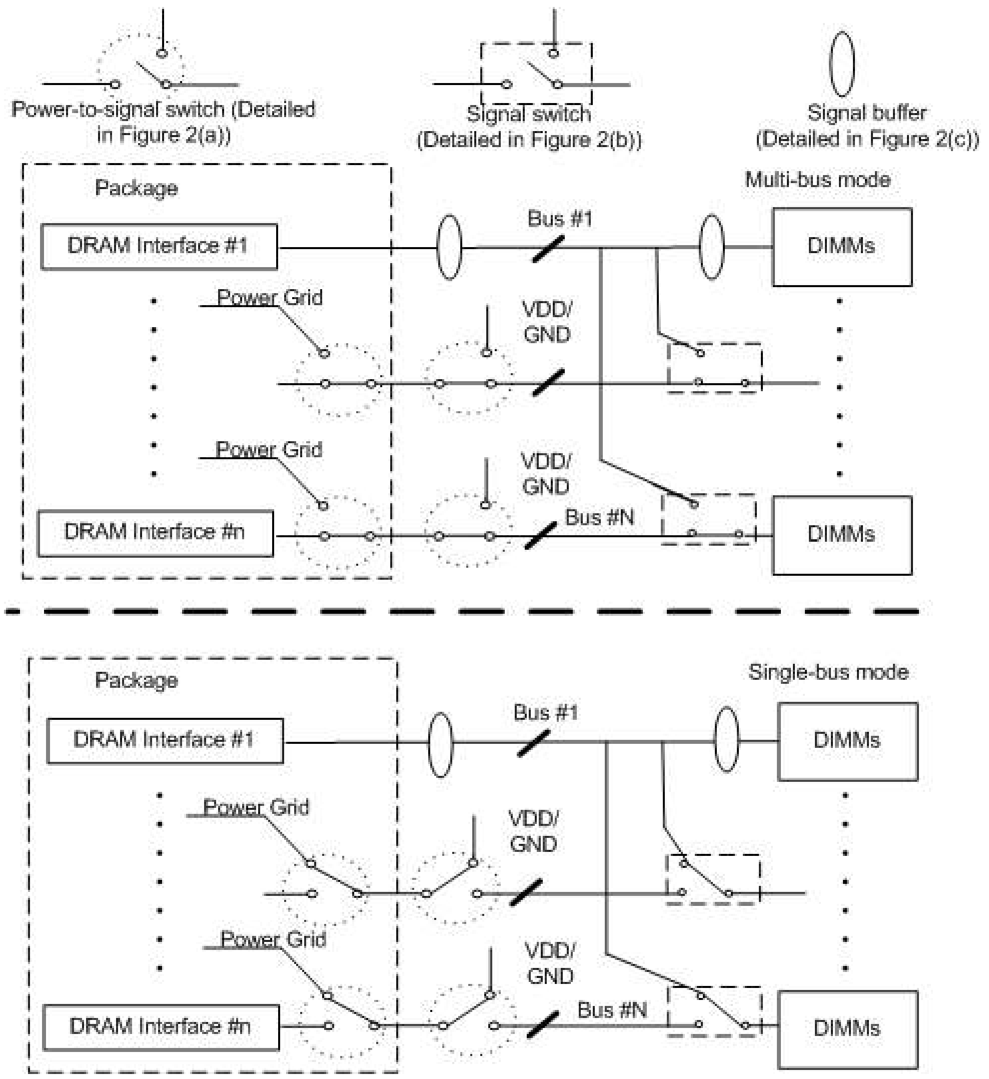


Figure 2-2. The overview of the hardware design of off-chip bus connection for switching between the Multi-bus mode and the Single-bus mode

Table 2-1. Pin allocation of an Intel Processor i5-4670

V_{DD}	GND	DDR3	Others	Total
153	475	250	272	1150

order to shorten the signal wire which has to bear high current for power delivery. To avoid signal reflections caused by an impedance mismatch, we keep the width of the signal wires and conduct an experiment to test the feasibility of high current via these signal wires. Based on a specification from the PCB manufacturer [9] and the DDR3 PCB layout guidelines [8], our simulation with COMSOL shows the MTTF of the 6mil signal wire could be more than 2.5 x 105 hours with a 1A current. On the other hand, the signal switch should be placed near the corresponding DRAM device to reduce signal reflections.

2.1.3 Memory Controller

The data availability of the memory controller is our primary concern. All the available memory buses in the multi-bus mode must be fully utilized to achieve maximum bandwidth while still allowing all the data in single-bus mode to be accessed. Due to the complicated synchronization of memory requests between memory controllers, the switch between the two bus modes is only implemented inside the memory controller. Within a memory controller, a memory interface is designed for each bus to fully exploit the benefit of the multi-bus mode without the interference of traffic from other buses compared to the design of multiple buses sharing a single memory interface.

The memory controller in our design includes dedicated request queues which buffer the incoming requests to the buses shown in Figure 2-3. Queues individually receive the requests from the front arbiter which employs its address mapping policy when dispatching requests. Once the requests are residing in the queues, they are fetched by the back arbiter. While in multi-bus mode, the requests are fed into their corresponding buses via the corresponding DRAM interfaces. Because memory interfaces can operate independently and in parallel, the memory bandwidth can be amplified by a factor of the number of memory buses. In the single-bus mode, the memory controller works similar to a conventional processor and communicates with the attached DIMMs as appended ranks.

2.1.4 Area Overhead

The circuit overhead of our design consists of the front arbiter, the end arbiter, and extra DRAM interfaces. As a result of both arbiters, the cost of dispatching requests without buffering them should be negligible. Furthermore, the cost of the additional DRAM interface is inexpensive. The estimated net area of a typical DRAM interface from Opencore [1] is 5,134 μm2 in 45 nm technology. This estimation is conducted by the Encounter RTL Compiler [5] with the NanGate Open Cell Library [6]. No more than

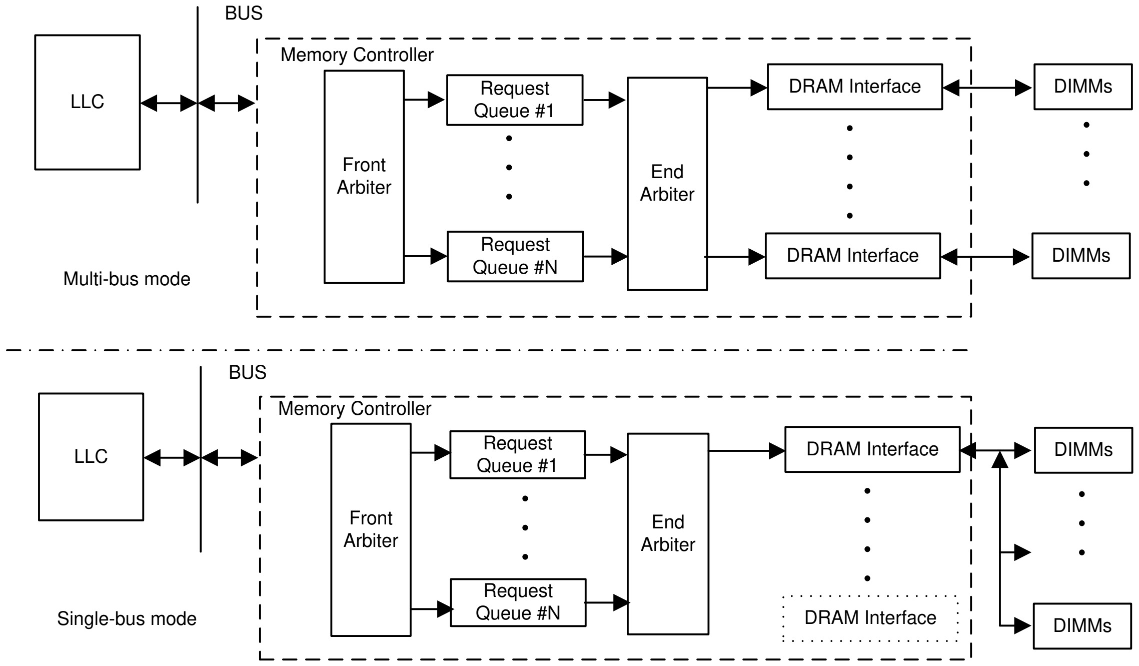


Figure 2-3. The Overview of the hardware design of memory controller for switching between the Multi-bus mode and the Single-bus mode

three additional buses in total are used in our experiment thus creating a maximum hardware overhead less than 0.00015 cm2 which is significantly less than the typical 1 cm2 die area.

2.1.5 Address Mapping

Data accesses interleave at the page level via different buses exploiting the benefit of memory-level parallelism while maintaining a high row buffer hit ratio. Interleaving at the block level considerably decreases the row buffer hit ratio resulting in longer off-chip latency per request and extended queue delay. To reduce row-buffer conflicts, we employ XOR banking indexing which could effectively reduce bank conflicts resulting from resource-contention-induced traffic and write-backs. This permutation distributes the blocks stored in the last level cache into different banks as opposed to possibly including tags of physical addresses containing the same bank index.

2.1.6 Signal Integrity

Signal integrity is analyzed to demonstrate feasibility in the single-bus and the multi-bus modes. We simulate SPICE models of our accessory circuit as well as PCB transmission lines, bond wire inductance, and driver capacitance associated with the device package in the AMS packages of Mentor Graphic as shown in Figure 2-4. The parameters are derived from previous works [58][62]. Signal integrity challenges are alleviated since the DDR3 command signal is unidirectional and its speed is no more than that of the data signals [58]. In this study, we only analyze the effect of our accessory circuit on the data signals which could be viewed as the worst case for all the signals.

In Figure 2-5 (a-d), the eye patterns of writing data (controller to device) and reading data (device to controller) in the two modes are derived from the corresponding SPICE models in Figure 2-4 (a-d) respectively. They have clear eyes since the signal-to-power switch alleviates the effect of the parasitic capacitance of the power switches. Furthermore, the signal switches as well as signal buffers alleviate the signal reflections caused by discontinuities. Thus, the results indicate our accessory circuit could maintain the signal quality in the two modes.

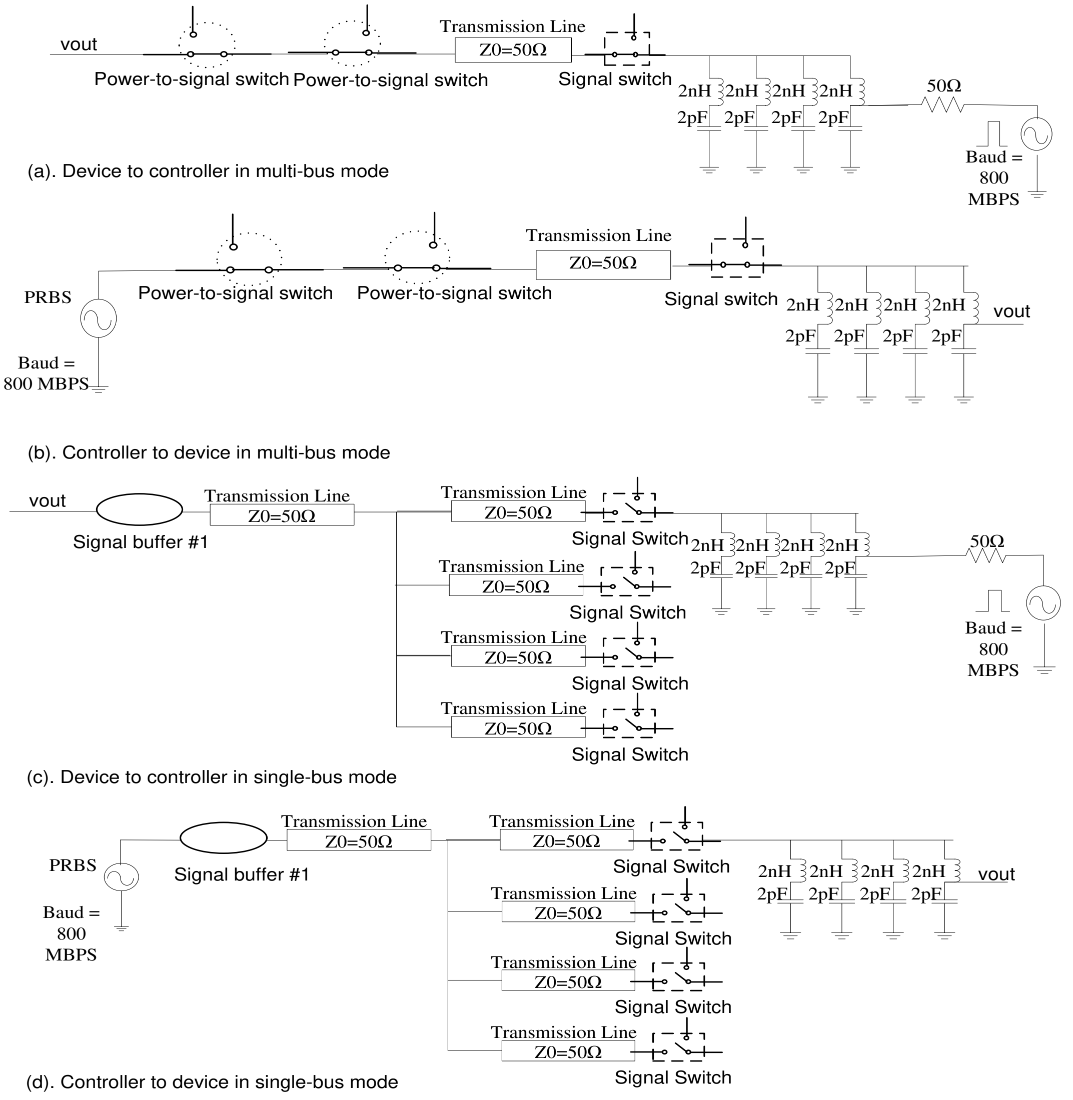
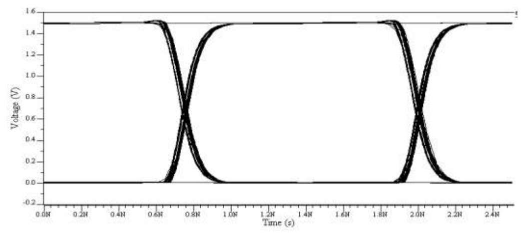
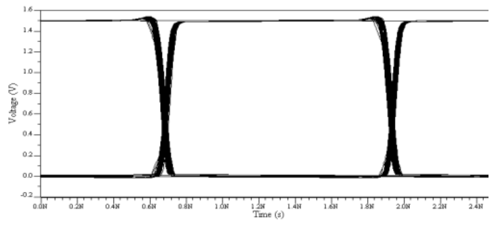
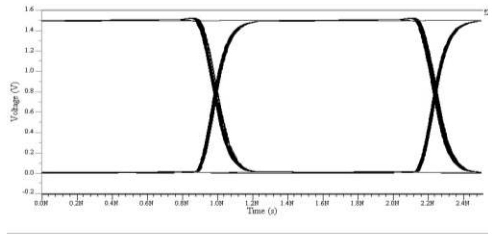

vout
Power-to-signal switch
Power-to-signal switch
Transmission Line
Z0=50Ω
Signal switch
2nH
2pF
50Ω
Baud = 800 MBPS
(a). Device to controller in multi-bus mode
PRBS
Baud = 800 MBPS
vout
(b). Controller to device in multi-bus mode
Signal buffer #1
Signal Switch
(c). Device to controller in single-bus mode
(d). Controller to device in single-bus mode

Figure 2-4. Spice models for signal integrity simulation


(a) DQ Read in multi-bus mode (Device to Controller)


(b) DQ Write in multi-bus mode (Controller to Device)


(c) DQ Read in single-bus mode (Device to Controller)

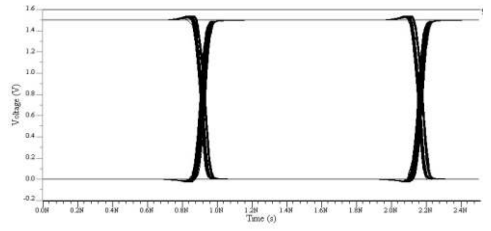

(d) DQ write in single bus mode (Controller to Device)

Figure 2-5. The eye diagrams

2.1.7 Power Delivery Simulation

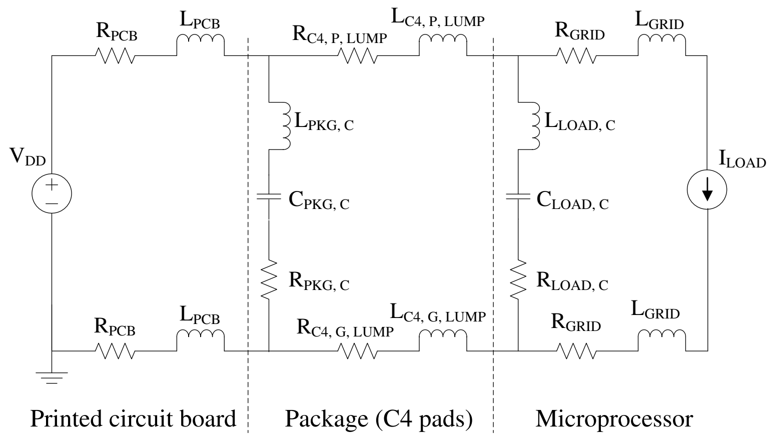
In this section, we assess the repercussions experienced by the power delivery network (PDN) when the switchable pins are shifted from single-bus mode to multi-bus mode. The PDN is depicted in Figure 2-6 (a). The power delivery path is modeled with RL components (i.e. resistors and inductors) connected in series across the PCB, the package, and the silicon die. Decoupling capacitors are introduced between each individual PDN to control any voltage fluctuations. The on-chip power grids and processor circuits on the silicon die are modeled separately as RL components with an ideal current source.

Figure 2-6 (b) illustrates the RL model of the Controlled Collapse Chip Connection (C4) pads [31] in which the resistance of the on-state power switches is taken into consideration. Table 2-2 lists the parameter values obtained from prior work [45].

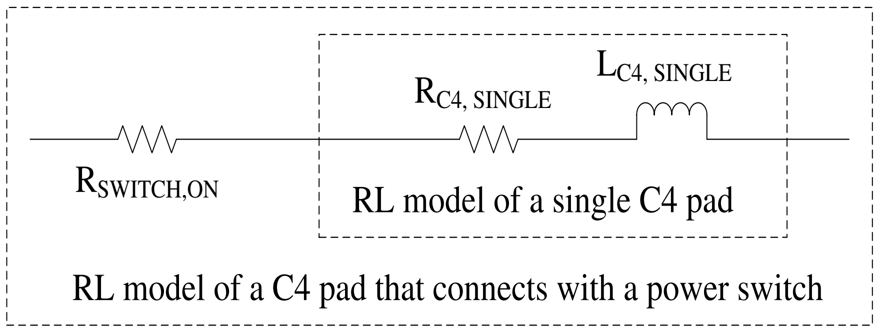
PDN simulations are performed in PSPICE to evaluate the impact of Pin Switching. Due to resistance along the power delivery path, an IR drop exists between the supply voltage and load voltage as current flows through the PDN. We assume a normalized IR drop should be upper-bounded by 5% as prior work dictates [52][56]. This implies that the maximum currents are 125A, 104A, 80A, and 56A for the baseline and then for Pin Switching mechanisms with one, two, and four borrowed buses respectively. In other words, the three Pin Switching diagrams switch 125, 250, and 375 power pins to signal pins providing 16.8%, 36.0%, and 55.2% less current with 19.9%, 39.8% and 59.7% less power pins respectively. The percentage of current decrease is less than that of

Table 2-2. Power network model parameters

Resistance	Value	Inductance	Value
R_{PCB}	0.015 mΩ	L_{PCB}	0.1 nH
$R_{PKG,C}$	0.2 mΩ	$L_{PKG,C}$	1 pH
$R_{LOAD,C}$	0.4 mΩ	$L_{LOAD,C}$	1 fH
R_{GRID}	0.01 mΩ	L_{GRID}	0.8 fH
$R_{C4,SINGLE}$	40 mΩ	$L_{C4,SINGLE}$	72 pH
$R_{SWITCH,ON}$	1.8 mΩ		
Capacitance			
$C_{PKG,C}$	250 μF	$C_{LOAD,C}$	500 nF



(a) Power delivery network


$R_{C4, SINGLE}$
$L_{C4, SINGLE}$
$R_{SWITCH,ON}$
RL model of a single C4 pad
RL model of a C4 pad that connects with a power switch

(b) RL model of a C4 pad

Figure 2-6. RLC power delivery model

proportional power pin quantity decrease because the IR drop depends on the resistance in the PCB and power grids.

We assume the processor employs a dynamic voltage and frequency scaling (DVFS) mechanism supporting 4 voltage and frequency operating points. The frequency can be scaled down from 4.0GHz to 1.2GHz. Correspondingly, the voltage will be decreased from 1.0V to 0.64V. According to McPAT [56], the baseline design can work at a frequency of 4.0GHz given the power delivery information. However, the processor frequency must be decreased individually to 3.2GHz, 2.4GHz, and 1.2GHz when the power pins for one, two, and three sets of memory channel pins are borrowed as I/O pins respectively. The results shown in Table 2-3 are used in the following evaluation.

Table 2-3. Processor power and frequency parameters for different number of buses

BUS	1	2	3	4
Current (A)	125	104	80	56
Voltage (V)	1	0.88	0.76	0.64
Power (W)	125	92	61	36
Frequency (GHz)	4	3.2	2.4	1.2

2.1.8 Runtime Switch Conditions

Designing a predictor to choose the most beneficial mode for the next interval is non-trivial for multi-program workloads. Simply switching based on the amount of consumed off-chip bandwidth is not sophisticated enough to improve the overall performance of a system in which only some of the programs that suffer from long off-chip access latency are likely to benefit from multi-bus mode. To identify intervals that will benefit from Pin Switching it is necessary to estimate both the performance change of each program and the overall benefit of switching for the following interval based on the current performance before a switching occurs. We introduce a metric called the switching benefit $B_{ij}(T_c)$ to help identify the most beneficial mode for each 1 millisecond interval, where $B_{ij}(T_c)$ represents the estimated reward for running the interval following time T_c in mode j instead of mode i. Based on the history of the switching benefit, we predict $\widetilde{B}_{ij}(T_c)$ as the switching benefit for the following interval using $\widetilde{B}_{ij}(T_c) = \sum_{k=1}^{N} B_{ij}(T_c - k * T_{interval})$, where $B_{ij}(T_c - k * T_{interval})$ represents the switching benefits detailed in equation (1) and can be measured from the N intervals ago and N is the length of the history to consider which were carefully chosen to be 2 for our experiment. If the predicted switching benefit is negative, the system will stay in mode i, otherwise, it will switch to mode j.

The switching benefit is calculated using the following equation:

$$\mathrm{B_{ij}(T_c)} = \sum_{k=1}^{p}(\mathrm{WS_{j,k}(T_c)} - \mathrm{WS_{i,k}(T_c)}) \qquad 2\text{-}1$$

Where $\mathrm{WS_{i,k}(T_c)}$ and $\mathrm{WS_{j,k}(T_c)}$ stand for the estimated weighted speedups for program k at time $\mathrm{T_c}$ in mode i and mode j respectively, while p represents the number of simultaneously executing programs which is equal to 4 in our experiment. The weighted speedup of each program in mode i during the interval can be estimated based on the information derived from hardware counters and off-line profiling, since the system is running in mode i during the current interval. The weighted speedup is calculated as follows:

$$\mathrm{WS_{i,k}(T_c)} = \mathrm{T_{alone,i,k}(T_c)/T_{shared,i,k}(T_c)} \qquad 2\text{-}2$$

$$\mathrm{T_{alone,i,k}(T_c)} = \text{Committed Inst}_{\mathrm{alone,k}}\mathrm{(T_c)}/(\text{average IPS}_{\mathrm{alone,k}}) \qquad 2\text{-}3$$

where $\mathrm{T_{alone,i,k}(T_c)}$ stands for the execution time of the same instructions running without interference from co-runners and $\mathrm{T_{shared,i,k}(T_c)}$ denotes the execution time of a fraction of program k running with others during the current interval which is equal to the length of an interval (1 millisecond). Furthermore, $\text{Committed Inst}_{\mathrm{alone,i,k}}\mathrm{(T_c)}$ stands for the number of committed instructions during the interval following $\mathrm{T_c}$ of program k, directly derived from a hardware counter since it should be identical to the number when program k shares the main memory system with others. Average IPS obtained from off-line profiling denotes the average number of executed Instructions Per Second (IPS) when program k running alone. These values are used to approximate $\mathrm{T_{alone,i,k}(T_c)}$ based on the assumption that the IPS of each program is relatively steady when it runs alone, since an accurate estimation of $\mathrm{T_{alone,i,k}(T_c)}$ is challenging [65].

The estimation of the weighted speedup of each program in currently unused mode j is more difficult compared to that in current mode i, since we can only estimate the performance of mode j according to the information collected in mode i. The weighted speedup is calculated as follows:

$$\mathrm{WS}_{j,k}(T_c) = T_{alone,j,k}(T_c)/T_{shared,j,k}(T_c) \qquad 2\text{-}4$$

$$T_{shared,j,k}(T_c) = T_{on-core,j,k}(T_c) + T_{off-core,j,k}(T_c) \qquad 2\text{-}5$$

Where $T_{alone,j,k}(T_c)$ is identical to $T_{alone,i,k}(T_c\ )$ and $T_{shared,j,k}(T_c)$ represents the execution time of program k running with others in mode j. It can be divided into two parts based on whether the execution times vary with core frequency: $T_{on-core,j,k}(T_c)$ denotes the portion of the execution time spent inside the core which is inversely proportional to core frequency, while $T_{off-core,j,k}(T_c)$ expresses the portion of execution time incurred by activity outside the core. We estimate $T_{on-core,j,k}(T_c)$ based on the corresponding time $T_{on-core,i,k}(T_c)$ in mode i using:

$$T_{on-core,j,k}(T_c) = T_{on-core,i,k}(T_c) * {}^{freq_{i,k}}/_{freq_{j,k}} \qquad 2\text{-}6$$

Where $freq_{i,k}$ and $freq_{j,k}$ are the frequencies in mode i and mode j respectively. We estimate $T_{on-core,i,k}(T_c)$ with the same breakdown using

$$T_{on-core,i,k}(T_c) = T_{interval} - T_{off-core,i,k}(T_c) \qquad 2\text{-}7$$

$$T_{off-core,i,k}(T_c) = T_{LLC,i,k}(T_c) + T_{DRAM,i,k}(T_c) \qquad 2\text{-}8$$

where $T_{LLC,i,k}(T_c)$ is the execution time incurred in the shared last level cache (LLC) in mode i, which is estimated using the number of the accesses to LLC, and $T_{DRAM,i,k}(T_c)$ denotes the execution time incurred by activity in the DRAM controller in mode i. $T_{DRAM,i,k}(T_c)$ is the cumulative time spent when there is at least one in-flight read requests in the DRAM controller, since it can avoid the overestimation due to the overlap of multiple in-flight read requests for single thread [68].

On the other hand, $T_{off-core,j,k}(T_c)$ is mainly affected by the number of buses between different modes since the queue delay inside the DRAM controller is typically decreased as more off-chip buses are added. We calculate the time using:

$$T_{off-core,j,k}(T_c) = T_{off-core,i,k}(T_c) + T_{queue\ delay,j,k}(T_c) - T_{queue\ delay,i,k}(T_c) \qquad 2\text{-}9$$

$$T_{queue\ delay,j,k}(T_c) = T_{queue\ delay,i,k}(T_c) * {}^{N_{request,j,k}(T_c)}\!\big/_{N_{request,i,k}(T_c)} \qquad 2\text{-}10$$

where $T_{queue\ delay,i,k}(T_c)$ and $T_{queue\ delay,j,k}(T_c)$ denote the execution time incurred inside the queue of the DRAM controller in modes i and j respectively, while $N_{request,i,k}(T_c)$and $N_{request,j,k}(T_c)$ stand for the average number of waiting requests per incoming read requests which have to wait until they have been completed in modes i and j. $T_{queue\ delay,i,k}(T_c)$ can be estimated by the time when there is at least one read request in the queue of DRAM controller. $T_{queue\ delay,j,k}(T_c)$ can be estimated by sampling the number of waiting requests in different modes

2.1.9 Switching Overhead

Any runtime overhead incurred by switching comes from the DVFS and IR drop fluctuations caused by the pin switch. The overhead for DVFS is 20μs [57] and the time for the IR drop to re-stabilize is also bounded by 20μs according to our power delivery simulation. Because both of these delays overlap each other, the estimated total overhead is 20μs and is taken into consideration. Therefore, the penalty is 40μs when a phase is incorrectly identified. However, the overall switching overhead is still negligible since the average length of the identified phases shown is much longer than the overhead in our workloads. Since most programs only switch a few times during execution, nearly all the program phase transitions have been identified by the predictor.

2.2. EXPERIMENTAL SETUP

To evaluate the benefit of our design, we simulate the x86 system documented in Table 2-4 using the Gem5 simulator [24]. We modify the DRAM model integrated in Gem5 to accurately simulate the proposed method. Throughout the experiments, multi-bus mode will utilize all available buses with the corresponding core frequency shown in Table 2-3. The buses are partially unutilized with a high core frequency between multi-bus and single-bus modes. We employ off-chip DVFS to maintain the same frequency on all 4 cores at any given time.

2.2.1 Performance and Energy Efficiency Metrics

We use weighted speedup [79] lists as follows to represent the throughput of our system shown in the following equation.

Table 2-4. The Configuration of the simulated system

Processor	4 X86 OoO cores with issue width 4
L1 I cache	Private 32KB, 8 way, 64B cache line, 2 cycles
L1 D cache	Private 32KB, 8 way, 64B cache line, 2 cycles
L2 Cache	Shared 8MB, 8 way, 64B cache line, 20 cycles
Memory controller	FR-FCFS scheduling, open row policy
Channel	1
Bus per channel	2 /3/4 (additional buses 1/2/3)
Rank per bus	2
Bank per rank	8
Bank	8*8 DDR3-1600 chips from Micron datasheet[62]

$$\text{Weighted Speedup} = \sum_{i=0}^{N-1} \frac{1/T_i^{\text{Shared}}}{1/T_i^{\text{Alone}}} \qquad \text{2-11}$$

where T_i^{Shared} and T_i^{Alone} denote the execution time of a single program running alone and the execution time running with other programs respectively. Because the IPC is distorted by the frequency change from the employed DVFS, the execution time is used in place of it. We utilize Energy per Instruction (EPI) for the evaluation of energy efficiency. This metric can be obtained from dividing consumed energy by the number total number of instructions committed.

2.2.2 Workloads

Various multi-program workloads consisting of SPEC 2006 benchmarks [76] are used for our evaluation. As listed in Table 2-5, the benchmarks are categorized into two separate groups based on their relative memory intensities: memory-intensive programs and compute-intensive programs. Each workload consists of four programs from one of these groups to represent a memory-intensive workload or compute-intensive workload accordingly. Memory-intensive workloads are used to demonstrate the benefit of multi-bus mode while the compute-intensive workloads demonstrate that there are negligible side-effects.

We select a simulated region of 200 million instructions for each benchmark based on their memory characteristics collected from Pin [10]. The simulation for a mixed workload does not end until the slowest program finishes its 200 million instructions. Faster programs continue running after committing the first 200 million instructions. Execution time of each program is collected after the program finishes its instructions.

2.3. RESULTS

The execution latency of a program is composed of the on-chip and off-chip latency. The percentage of latency in the total execution time reveals which factor tends to be more influential to the overall performance of a workload. In Figure 2-7 we demonstrate the off- chip latency for memory-intensive workloads and on-chip latency for the compute-intensive workloads, since they are the main contributors to the execution latency of the two categories of workloads, respectively. Specifically, more than 80% of the latency of memory-intensive workloads comes from off-chip latency,

Table 2-5. The selected memory-intensive and compute-intensive workloads

workload				
Memory-intensive programs				
M1	lbm	milc	soplex	libquantum
M2	lbm	milc	leslie3d	libquantum
M3	lbm	milc	soplex	leslie3d
M4	lbm	soplex	libquantum	leslie3d
M5	milc	soplex	libquantum	leslie3d
M6	mcf	mcf	mcf	mcf
M7	mcf	mcf	astar	astar
M8	astar	astar	astar	astar
Mixed programs				
MIX1	lbm	milc	bzip2	bzip2
MIX2	lbm	milc	omnetpp	omnetpp
MIX3	lbm	soplex	omnetpp	omnetpp
MIX4	milc	soplex	omnetpp	omnetpp
MIX5	lbm	milc	omnetpp	bzip2
MIX6	milc	soplex	omnetpp	bzip2
Compute-intensive programs				
C1	bzip2	bzip2	bzip2	bzip2
C2	hmmer	hmmer	hmmer	hmmer
C3	gromacs	bzip2	omnetpp	h264ref
C4	gromacs	bzip2	sjeng	h264ref
C5	gromacs	omnetpp	sjeng	h264ref
C6	bzip2	omnetpp	sjeng	h264ref

while more than 60% of the latency of compute-intensive workloads is from on-chip latency. This implies that the memory-intensive workloads could be sped up by our Pin Switching, while the others are unlikely.

2.3.1 Memory-Intensive Workloads

Figure 2-8 shows the performance improvements of memory-intensive workloads enhanced by 2, 3, and 4 buses. The weighted speedup of each case is normalized against its own baseline. The baseline is the simulated system fixed in the single-bus mode with

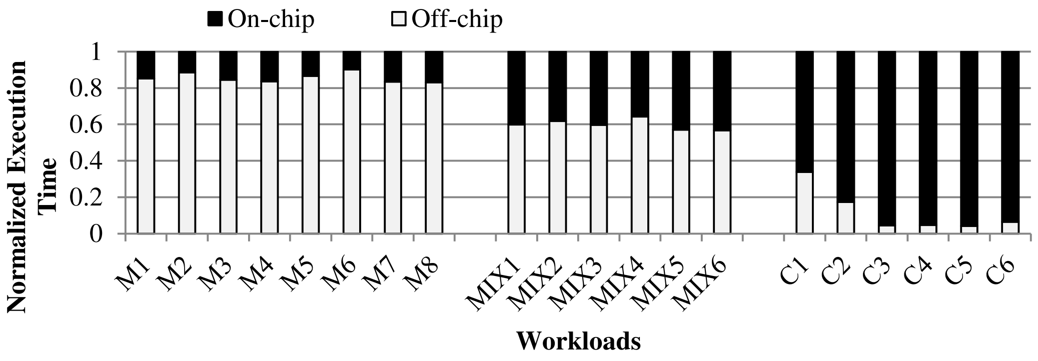


Figure 2-7. The normalized off-chip latencies and on-chip latencies of workloads against the total execution time

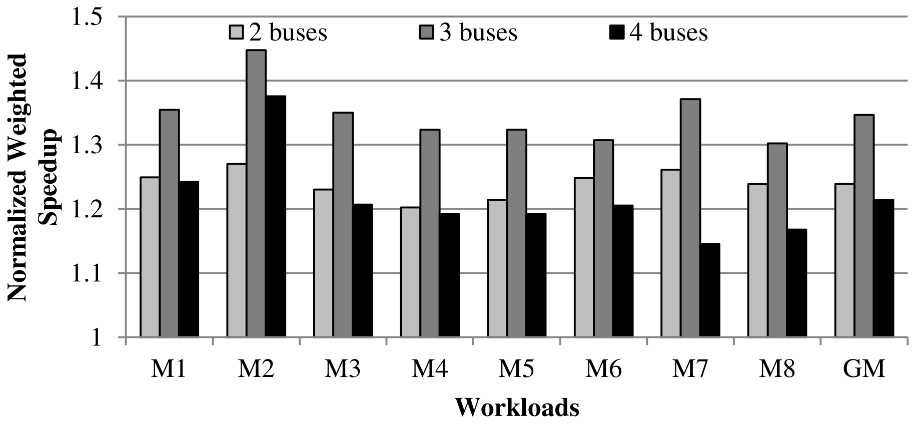


Figure 2-8. The normalized weighted speedup of memory-intensive workloads with 2, 3, and 4 buses against the each baseline

the corresponding number of buses and DRAM devices when the processor runs at 4.0GHz. Remarkably, the improvements experienced with 3 buses consistently surpass 2 and 4 buses in all workloads. These results stem from the balance between core performance and off-chip bandwidth that the 3 buses experience to maximize the throughput of the simulated system. Based on our specific hardware configuration and selected workloads, the multi-bus mode with 3 buses is the optimal choice and therefore referred to as the default configuration for the discussion of Static and Dynamic Switching that will be presented in later sections. Figure 2-9 illustrates the performance improvement for multi-bus mode tested using various DRAM configurations. The weighted speedup for each configuration is normalized against the same configuration in single-bus mode. As can be seen from the figure, all banks and ranks have weighted speedups greater than 32%. As the number of ranks per channel or the number of banks per rank increases, improvement is slightly diminished due to the resulting lower row buffer hit ratio causing shorter bank access latency.

Figure 2-10 presents the benefits of Static Switching and Dynamic Switching with 3 buses versus the baseline of a simulated system that does not use the pin switch mechanism on memory-intensive workloads. Both schemes are able to speed up the execution of all workloads by more than 1.3 times, while an approximately 42% performance improvement is observed for M2. The geometric means of Static Switching and Dynamic Switching are respectively 1.34 and 1.33 due to more than 99% of the running time being identified as typical memory-intensive phases by Dynamic Switching.

The benefit of the multi-bus mode is mainly attributed to the increase of consumed bandwidth as shown in Figure 2-11. The increase is similar to this of the

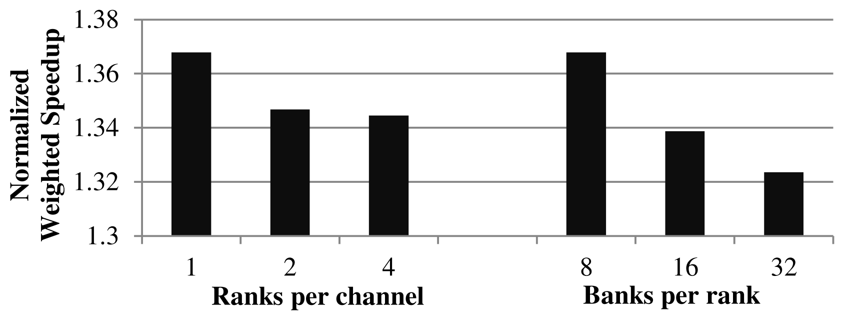


Figure 2-9. The average normalized weighted speedup of memory workloads in geometric mean with multi-bus mode. Each normalize to the same configuration with single bus mode

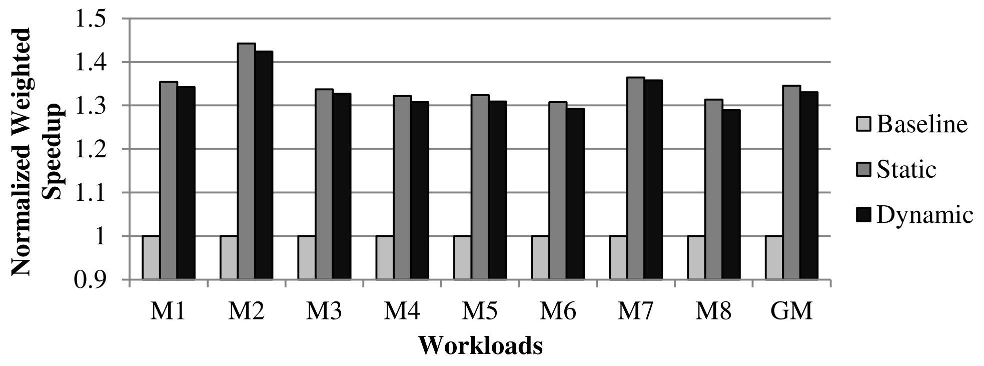


Figure 2-10. The normalized weighted speedup of memory intensive workloads boosted by Static Switching and Dynamic Switching with 3 buses against the baseline

weighted speedup in Figure 2-9. For example, M2 and M7 gain 47% and 39% off-chip bandwidth when switching from the single-bus mode to the multi-bus mode for static switching, while their performances are improved by 44% and 36% respectively. This similarity results from the fact that their execution latencies are largely dominated by off-chip latency. On the other hand, Dynamic Switching achieves a slightly smaller increase in bandwidth, which results in its performance being close to that of Static switching.

The throughput improvement of Dynamic Switching could be strengthened by using prefetchers which can utilize extra bandwidth brought by additional buses in our design. In our experiment, we use a stride prefetcher in the last level cache to demonstrate the benefit. More sophisticated prefetchers could be employed to further improve the system performance. The stride prefetcher used here has a prefetching degree of 1, 2, or 4, which denotes the number of prefetches issued on every memory reference. As illustrated in Figure 2-12, the geometric mean of the performance improvements of Dynamic Switching for all memory-intensive workloads with a prefetching degree of 1, 2, and 4 are 1.51, 1.64, and 1.79 respectively, compared with those of the baseline which are 1.10, 1.17, and 1.27. The gap of the improvements between Dynamic Switching and the baseline increases as the prefetch degree increases, which imply an aggressive stride prefetch could benefit more from Dynamic Switching.

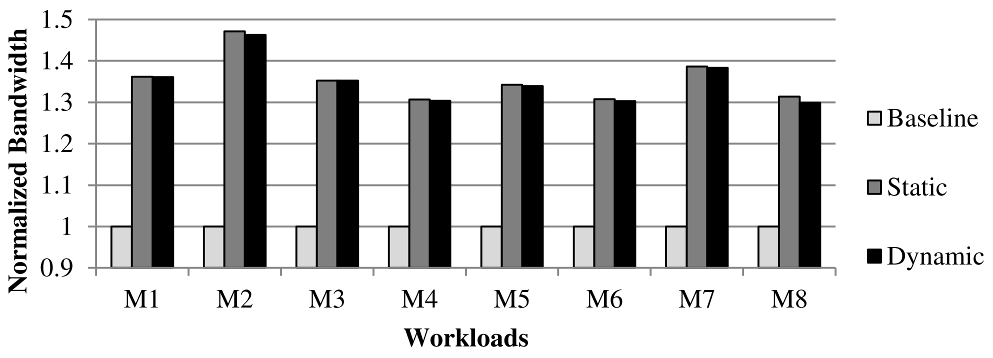


Figure 2-11. The increased bandwidth due to pin switching. The normalized bandwidth of baseline, static pin switching, and dynamic pin switching

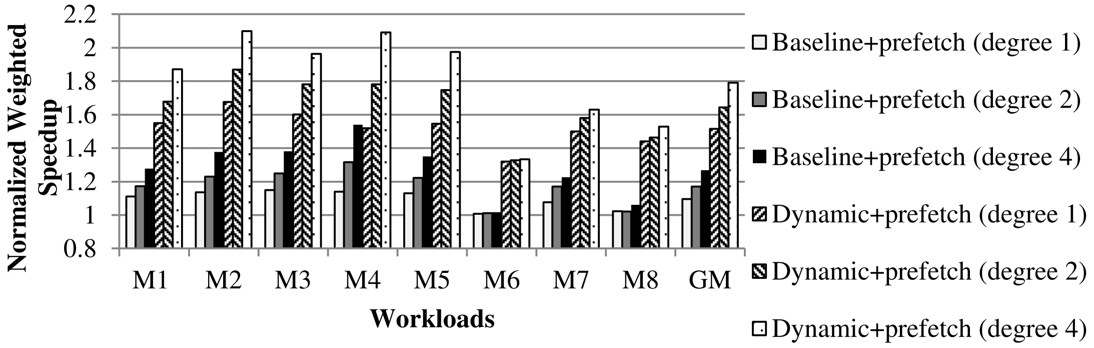


Figure 2-12. The improved throughput of Dynamic Switching boosted by a stride prefetchers (degree = 1, 2, 4) for memory-Intensive workloads

This observation could be demonstrated in all workloads except M6 which only gains a slight performance improvement from increasing the prefetch degree, since the stride prefetcher has a low coverage on mcf [42]. This performance improvement could be verified by the higher consumed off-chip bandwidth of Dynamic Switching shown in Figure 2-13. It implies that Dynamic Switching could boost the performance of the prefetch by providing more off-chip bandwidth.

The energy efficiency of the system could be also improved by Dynamic Switching. Figure 2-16 details the energy efficiency improvement of the simulated system. In theory, the energy savings come from two sources: (1) low voltage and frequency scaling; and (2) the execution reduction time stemming from multiple buses brought by pin switching. We quantify the first part by setting the core frequency of the simulated system to 2.4 GHz (relating to the frequency of our multi-bus mode scheme) with the corresponding voltage for single bus. The results depicted as gray bars in Figure 2-16 demonstrate 40% improvement in the geometric mean of the EPI for all the workloads over the baseline. Note that the overall execution time of this setting is only

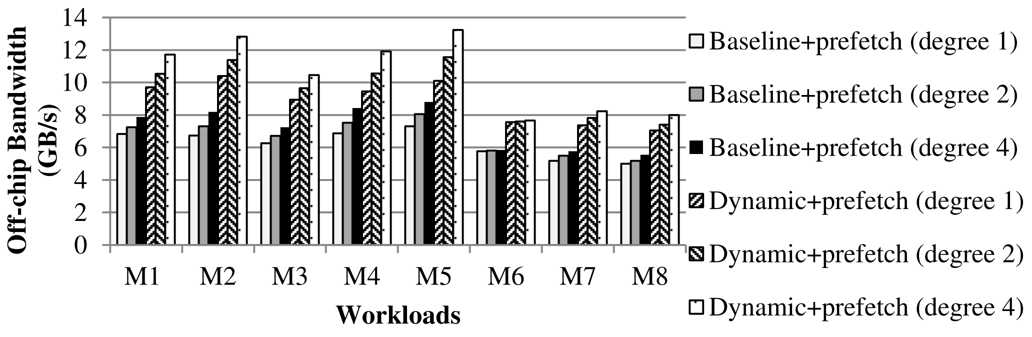


Figure 2-13. The off-chip bandwidth of Dynamic Switching improved by a stride prefetcher (degree = 1, 2, 4) for memory-Intensive workloads

slightly longer than that of the baseline system because all workloads are memory-intensive. Furthermore, the multi-bus mode offers an average of 66% improvement in the geometric mean of the EPI for all the workloads over the baseline resulting from execution time reduction.

2.3.2 Wide-bus mode

We also introduce wide-bus mode as another approach to increase off-chip bandwidth by using switchable pins compared to the multi-bus mode. Wide-bus mode uses switchable pins to widen the data bus to increase off-chip bandwidth. In wide-bus mode, DIMMs share the command and address bus but have dedicated data buses. Wide-bus mode only needs to alter the states of the signal-to-power switches and signal switches on the data buses. Thus, it increases the off-chip bandwidth at a low cost of switchable pins, since it can double the off-chip bandwidth of a 64 bit memory bus by using 64 switchable pins instead of 125 ones for a whole memory bus. Additionally, it incurs less overhead in the memory controller since it only needs a modified DRAM interface for moving data over the wider bus instead of extra DRAM interfaces. The challenge of implementing the wide-bus mode comes from keeping equal delays between all DIMMs and processor pins. It is solvable although it requires considerable efforts to route the traces connecting the DIMMs and the pins of processor.

Wide bus mode uses pins to widen the data path of memory buses instead of increasing the number of buses. Wide bus mode has two configurations: (1) the width of every memory bus is 128 bits and all cores are running at 3.6GHz; (2) the width of memory buses is 256 bits and all cores are running at 2.8 GHz. These configurations are calculated using the same method used for multi-bus mode.

Wide-bus mode is also tested in the simulated system with a memory bus and the three bus width configurations (64 bits, 128 bits, 256 bits). The three corresponding core frequencies for the bus widths are 4GHz, 3.6GHz, 2.8GHz derived based on the pin configuration. The baseline uses a bus width of 64 bits and a core frequency of 4GHz.

Figure 2-14 shows the performance improvement of wide-bus mode in two separate configurations: 128bit_3.6GHz in which the processor runs at 3.6GHz with a 128-bit memory bus; and 256bit_2.8GHz in which the processor runs at 2.8GHz with a 256-bit memory bus. 128bit_3.6GHz and 256bit_2.8GHz have a normalized weighted speedup in geometric mean of 1.1 and 1.15 respectively for memory intensive workloads. These moderate performance benefits are less than that of multi-bus mode especially for the M6 workload which consists of four instances of mcf. The M6 workload suffers from a high row buffer miss ratio and the resultant longer bank access latencies compared to

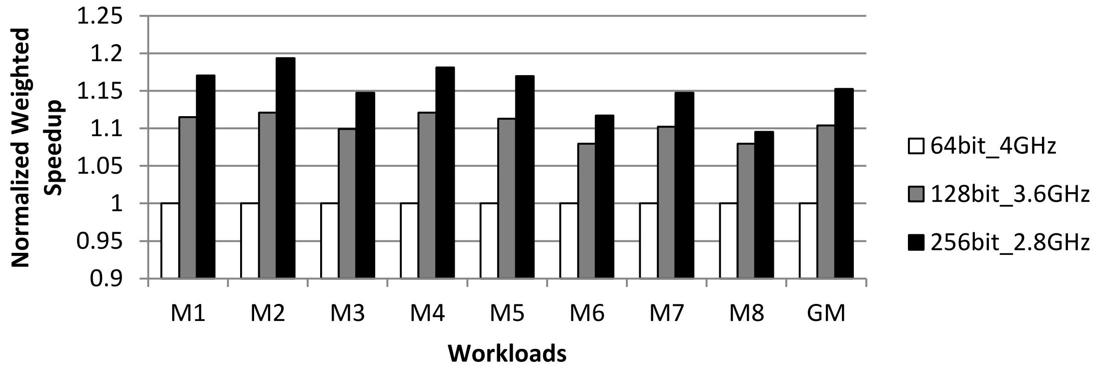


Figure 2-14. The performance of memory intensive workloads for the baseline (core frequency of 4GHz and a memory bus of 64 bits) and two configurations of wide bus mode (core frequency of 3.6GHz and a memory bus of 128 bits; core frequency of 2.8GHz and a memory bus of 256 bits).

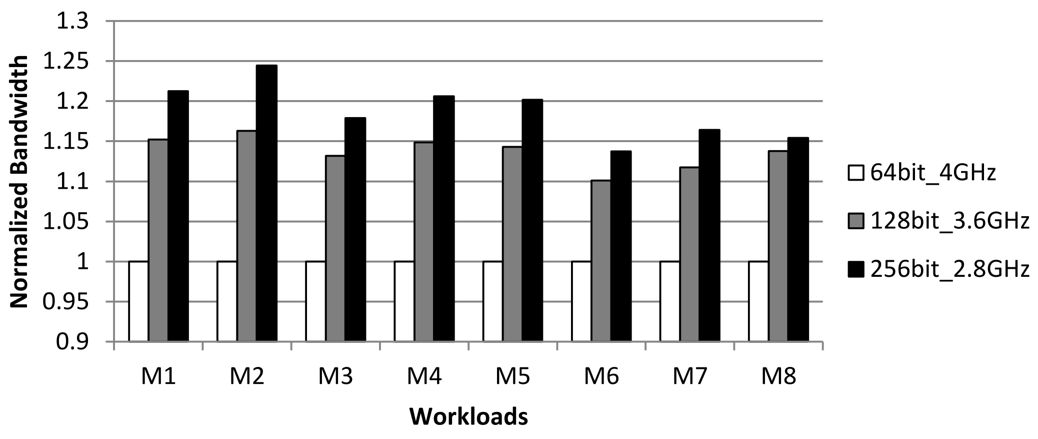


Figure 2-15. The off-chip bandwidth of memory intensive workloads for the baseline (core frequency of 4GHz and a memory bus of 64 bits) and two configurations of wide bus mode (core frequency of 3.6GHz and a memory bus of 128 bits, core frequency of 2.8GHz and a memory bus of 256 bits).

the latencies of moving the data over the bus. Since wide-bus mode only reduces bus latencies and cannot hide bank latencies, it delivers less performances benefits for this kind of applications. The increasing off-chip bandwidth in wide-bus mode presents a similar trend for the memory intensive workloads shown in Figure 2-15. In conclusion, wide-bus mode delivers less performance benefits compared to multi-bus mode. It only shortens the time of transferring data over the bus for a memory request while multi-bus mode hides the latencies of accessing banks and moving data over the bus by allowing multiple in flight memory requests. Thus, we prefer multi-bus mode over wide-bus mode for increasing off-chip bandwidth of processors in the following experiments.

2.3.3 Mixed Workloads

Figure 2-17 shows the system performance improvement of mixed compute-intensive and memory-intensive workloads using Pin Switching. The highest benefit is achieved using the 2 buses and per-core DVFS [83], which is the configuration used in

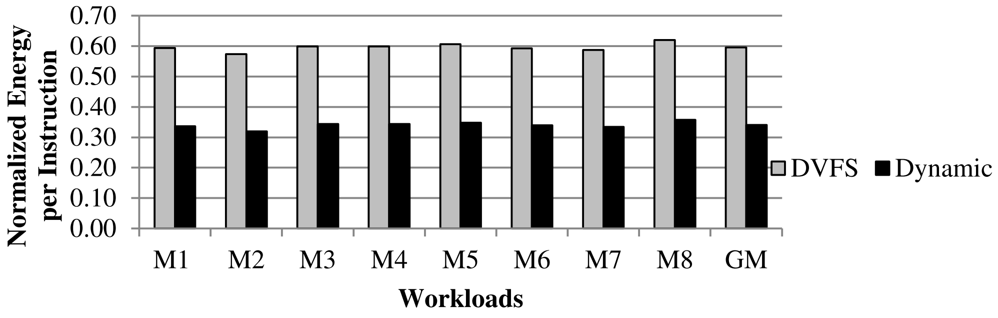


Figure 2-16. The normalized EPI of Dynamic Switching for memory intensive workloads with 3 buses, and the EPI from DVFS (running on 2.4GHz with the single bus)

this experiment after we explored the configuration space for these workloads. . The geometric means of the normalized weighted speedup from using Static Switching and Dynamic Switching are 1.10 and 1.09 respectively, implying that Dynamic Switching captures the most benefit of Pin Switching for these mixed workloads. Figure 2-18 shows the co-improvement of Pin Switching and stride prefetching with varying degrees (1, 2, 4) compared with the improvement of the prefetching alone. The geometric means of the normalized weighted speedup of Dynamic Switching with prefetching degree (1, 2, 4) are 1.15, 1.16, 1,15 respectively, while the means with prefetching alone are all 1.04. The co-optimization for all workloads saturates, or even slightly drops as the degree increases, which implies aggressive prefetching wastes off-chip bandwidth rather than exploiting the benefit of MLP for workloads. This can be confirmed by observing the performance of the baseline using prefetching alone as the degree increases.

2.3.4 Compute-Intensive Workloads

Figure 2-19 depicts the Dynamic Switching efficiency of compute-intensive workloads in comparison to Static Switching at the cost of lower core frequency and the base-line. The geometric mean of performance degradation for compute-intensive

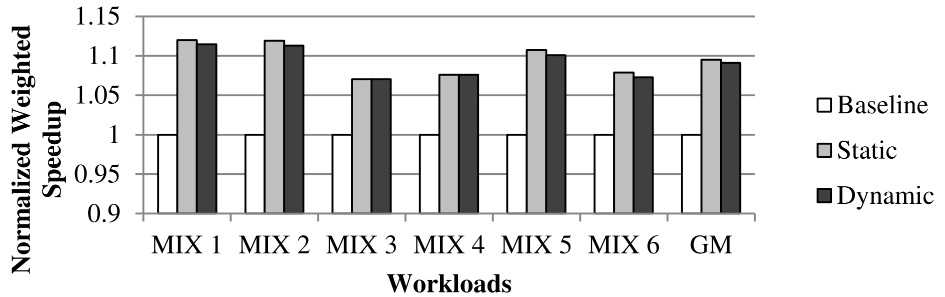


Figure 2-17. The normalized weighted speedup of mixed workloads boosted by Static Switching and Dynamic Switching

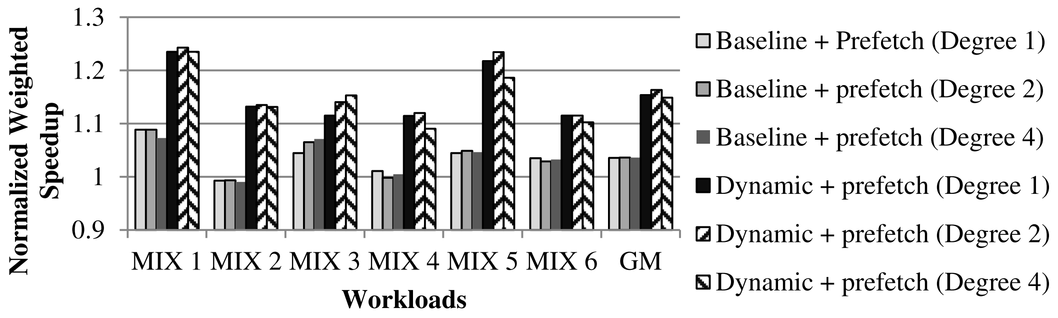


Figure 2-18. The improved throughput of Dynamic Switching boosted by a stride prefetchers (degree = 1, 2, 4) for mixed workloads

workloads introduced by the Static Switching scheme is 29%. The worst case results in a 35% slowdown of C5. In contrast, Dynamic Switching retains the same performance as the baseline during compute-intensive workloads because our metric successfully identifies non-memory-intensive phases when the rewards of the multi-bus mode are limited. Furthermore, Dynamic Switching surpasses the baseline for the C1 workload by identifying compute-intensive and memory-intensive phases. Overall, Dynamic Switching exhibits no performance penalty on compute-intensive workloads, in contrast to Static Switching.

The energy consumption of the Dynamic Switching mechanism is almost the same as the baseline since the processor runs at single-bus mode most of the time for

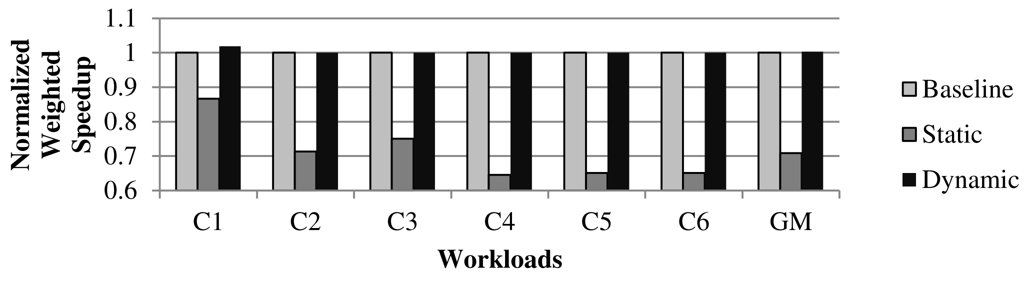


Figure 2-19. The normalized weighted speedup of Compute-Intensive workloads with Static Switching and Dynamic Switching

compute- intensive programs. Therefore, we do not illustrate the EPI comparison figure here.

2.4. RELATED WORK

DRAM-Based Memory System: Several papers propose to physically alter the main memory in a DRAM-based memory system to improve the performance and energy efficiency. Zhang et al. propose setting the bus frequency higher than the DRAM module to improve channel bandwidth where the induced bandwidth mismatch is resolved by a synchronization buffer inside the DIMM for data and command [93]. Papers also explore using low power DDR2 (LPDDR2) memory, in place of conventional DDR3, due to its higher energy efficiency [58][88].

To reduce the delay of bank access, thereby increasing memory bandwidth, architects optimize the memory system at the rank and bank level. Zhang et al. subdivides conventional ranks into mini-ranks with a shorter data width. These mini-ranks can be operated individually via a small chip on each DIMM for higher DRAM energy efficiency [94]. Rank sub-setting is also proposed to improve the reliability and performance of a memory system [19].

Inside a DRAM bank, increasing the row buffer hit ratio is key to improving energy efficiency and performance. Kim et al. partition a row buffer into multiple sub-arrays inside a bank to reduce the row buffer miss rate [47]. An asymmetric DRAM bank organization can be used to reduce the bank access latency and improve the system performance [85]. Unlike preceding work, we focus on increasing off-chip bandwidth to boost the performance of the memory system since it is the major bottleneck of memory systems in the multi-core era.

Off-Chip Bandwidth: Rogers et al. have already stressed the significance of off-chip bandwidth [73]. To increase the overall energy efficiency of a memory system, Udipi et al. split a 64 bit data bus into eight 8 bit data buses reducing the queue delay at the expense of data transfer delay [86]. Ipek designs a memory scheduler using principles of reinforcement learning to understand program behaviors and boost performance [43]. Mutlu et al. focus on boosting multi-threaded performance by providing fair DRAM access for each thread in their memory scheduler [65][66]. Our method of adding additional buses to multiply the off-chip bandwidth is orthogonal to the aforementioned methods, which focus on the memory scheduler and bus control.

Tradeoff between core performance and off-chip bandwidth: Architects employ several sophisticated methods to balance core and memory performance [23][29][32]. However, few of them are able to increase the off-chip bandwidth beyond the constraint of static pin allocation

CHAPTER 3. MITIGATING DARK SILICON VIA SWITCHABLE PINS

3.1. BACKGROUND

Integrated circuit (IC) packaging is the final process of the IC fabrication in which the silicon die (i.e., the core of the device) is encased in a support and connected to the chip package for power delivery and off-chip communication. There are two main technologies for connecting the silicon die with the chip package: wire bonding and flip chip. Wire bonding uses bonding wires to connect the pads located on the perimeter of the silicon die to the package. The flip chip technology, also called Controlled Collapse Chip Connection (C4) technology, is shown in Figure 3-1. The silicon die faces downwards, and is connected to the substrate directly with C4 pads. C4 technology greatly increases pad density, compared with wire bonding, by allowing C4 pads to be placed over the entire chip area. This eases wiring requirements by allowing shorter wire lengths and fewer global wires, and provides better power distribution as circuits in the middle of the die can access VDD/GND directly. The size of a silicon die is smaller than

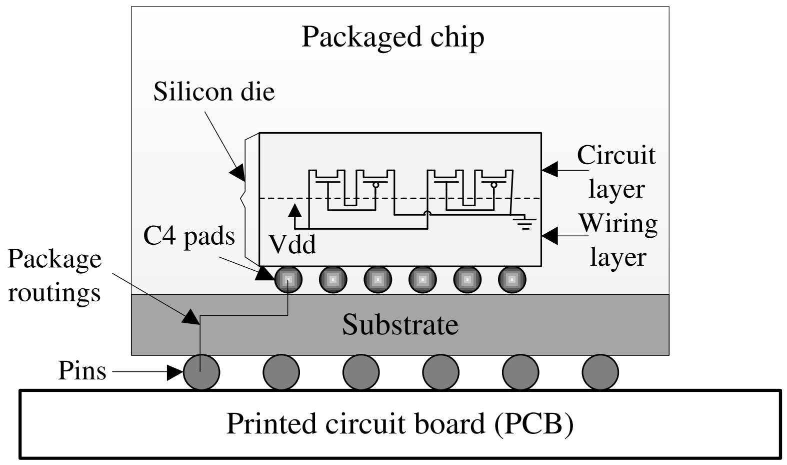


Figure 3-1. Structure of a packaged chip (8 copies of DEALII from SPEC2006)

that of the chip package. This means the cross-sectional area of a C4 pad is smaller than that of a pin as shown in Figure 3-1. According to a recent study [90], it is concluded that I/O pad shortage will limit power delivery in future sub-16nm technology. In addition, increasing the number of C4 pads will linearly increase chip packaging costs, which have already started to exceed the silicon fabrication costs [44].

3.2. OVERVIEW DESIGN

We now discuss how the computer system functions while utilizing switchable pins to deliver power. Figure 3-2 shows an overview of the dynamic pin switching design illustrating the layout of the microprocessor and SDRAM on the motherboard. The 64-bit data path of the integrated memory controller in the microprocessor connects to the SDRAM via 64 pins, specifically 16 conventional pins and 48 switchable pins. The 16 conventional pins are always used as I/O pins, while the switchable pins can switch between power pins and I/O pins dynamically. Our COMSOL-based [14] simulation

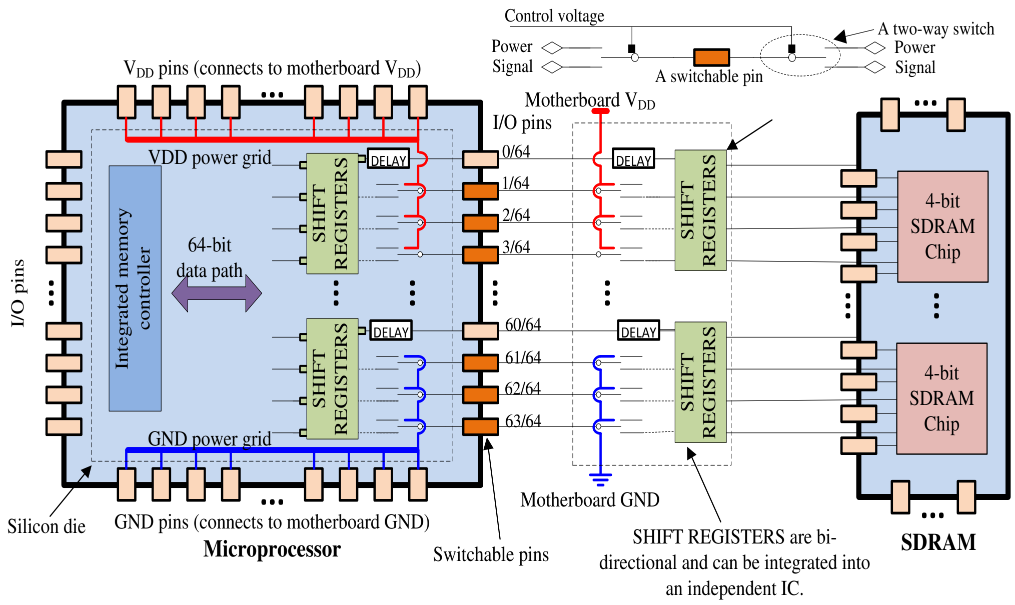


Figure 3-2. Design overview on the proposed scheme

which models the electromigration phenomenon on the traces/interconnects shows that using wires connecting to I/O pins to deliver the current studied in this work will not result in reliability issues. When the control voltage is low, the computer system works in the default I/O mode since the switchable pins are used as I/O pins. In this mode, signals circumvent the shift register components on the microprocessor and motherboard, which causes the 64-bit data path of the integrated memory controller in the microprocessor to connect to the SDRAM via 64 I/O pins directly. On the other hand, when the control voltage is pulled up the switchable pins are used as power pins; thus the computer system works in the power mode. In this mode, all shift register components are enabled to implement the signal transmission via the limited 16 I/O pins. The shift registers are bi-directional, one is parallel-in serial-out while the other is serial-in parallel-out, and have a negligible area overhead [2][3]. When switchable pins are used for signal transmission, shift registers steer the signal from input to output without buffering them. Otherwise, they are used to send signals over a single line instead of 4 lines. The shift registers can be integrated into the microprocessor and motherboard and synchronized by the clock signal of SDRAM interface. We also add a delay circuit to balance the delay between lines with and without signal buffers.

The shift registers work at the same frequency as the SDRAM and integrated memory controller. Therefore, in power mode it takes four times as many cycles to transfer data over the bus via 16 I/O pins as it does via 64 I/O pins. The equivalent bus frequency is decreased to 25% of its default value when the switchable pins are used for power delivery although only data I/O pins are influenced (i.e., the number of effective I/O pins is decreased from 64 to 16), which can reduce the bus power [30]. Although the design increases the time required for transferring data over the bus, it will not affect bank access time or the queuing delay.

To minimize the change to the computer system, we only consider one-way pin switching, i.e., dynamically allocating a portion of I/O pins to power pins. In fact, it is feasible to switch from power pins to I/O pins by designing extra I/O units (e.g. memory controllers) and related control logics. Switching from power pins to signal pins will increase the off-chip communication bandwidth, which boosts the performance of memory intensive workloads significantly. This work focuses on switching from signal pins to power pins since the major purpose is to find an approach to power up dark silicon.

3.2.1 Pin Allocation

To see how many switchable pins can be designed in a processor, we study the pin allocations of an Intel Xeon Processor E5-2450L [17] as listed in Table 3-1. We assume an equal number of C4 pads are designed on the chip with a pad density of 1356 pads/cm2 approximates to about 1200 pads/cm2 in the typical pad design [90]. Although it is feasible to design denser pads, the current that each pad can deliver will be smaller. The Xeon is an 8-core processor with a 20MB last-level cache and three memory channels. As can be seen, most pins are used for power delivery and off-chip communication. Among the off-chip communication pins, three 64-bit DDR3 memory channels occupy 483 C4 pins. Out of the pins on a 64-bit data path, 48 pins can be designed as switchable pins. Correspondingly, three memory channels have 144 switchable pins which can increase the number of power pins by 28.6% (i.e.,

Table 3-1. Pin allocation of the Intel Xeon Processor E5-2450L

V_{DD}	GND	DDR3	PCIE	QPI	DMI2	Others	Total
151	353	483	102	45	16	206	1356

144/(151+353)). On the other hand, among power delivery pins the number of the GND pins is more than that of the VDD pins. This helps lower the ground voltage in the silicon die, increasing circuit reliability, since the ground voltage is also used as a reference voltage for signal transmission. Conservatively following the same VDD/GND ratio, we allocate 144 switchable pins to 45 VDD pins and 99 GND pins in the pin switching mode. More switchable pins can be designed from other pins in DDR3 and pins in PCIE, QPI, DMI2 and etc. As an initial study, we only consider the 144 switchable pins from a portion of the data I/O pins in the three memory channels (DDR3).

3.2.2 Power Delivery Network

Here we study the impact on the power delivery network when the switchable pins switch from I/O mode to power mode. In the power delivery network (PDN) shown in

Figure 2-6 we assume the voltage regulator module is a fixed voltage source since its feedback control mechanism can maintain a steady output voltage regardless of current magnitude. The power delivery path across the printed circuit board (PCB), the package, and the silicon die are modeled as the RL (i.e., resistor and inductor) components connected in series. Decoupling capacitances are introduced between each sub power network to reduce the voltage bounce. Power grids and processor circuits of the silicon die are modeled separately as RL components and an ideal current source. Table 2-2 gives the parameter values obtained from prior works [45][50].

We perform static PDN simulations using SPICE [62]. There is an IR drop between the supply voltage and the load voltage as current flows through the PDN. As the total current increases, the IR drop increases due to the resistance on the power delivery path. We assume the normalized IR drop should be limited to be less than 5% as a design convention used by previous work [52][90] to ensure signal integrity and energy efficient power delivery. Thus, the maximum allowable currents are respectively 116A and 144A for the baseline and the pin switching design. In other words, the pin switching design can supply an extra 24.1% (i.e., (144-116)/116) current with 28.6% more power pins. The pin switching design can supply a larger current since it provides more power pins that reduce the package resistance. The percentage of current increase is less than that of power pin increase because the IR drop also depends on the resistance on the PCB and power grids. In addition, our processor power model shows that the extra current can boost the frequency of an 8-core processor from 2.0GHz to 3.0GHz in dim silicon mode. As listed in Table 3-2, the delivered power increases from 75.4W (0.65V×116A) to 111.6W (0.775V×144A) by 48.0% (i.e., (111.6-75.4)/75.4)). Note the supply voltage is

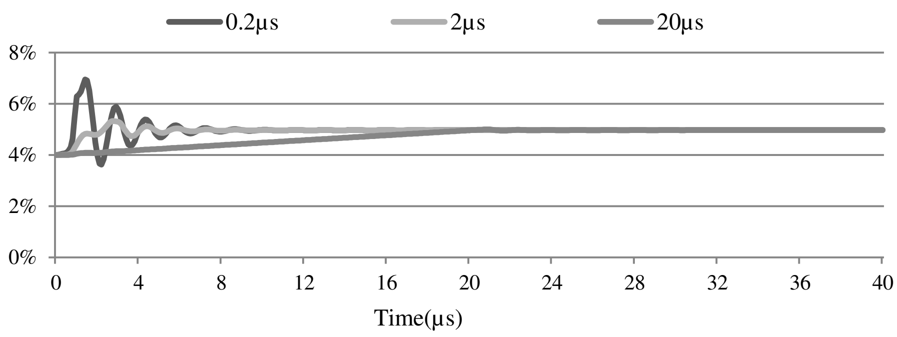


Figure 3-3. Dynamic simulation

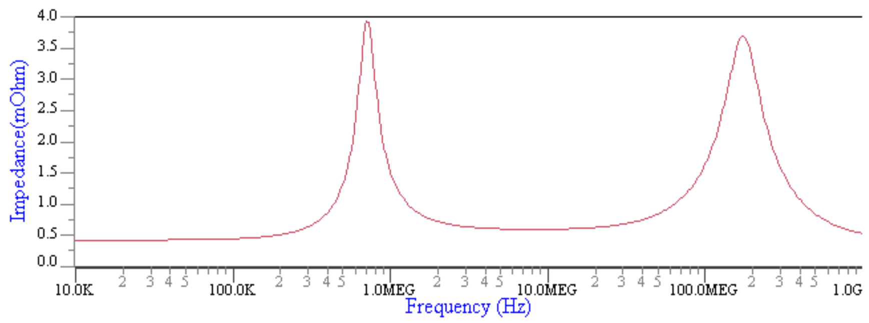
Table 3-2. Processor configurations under different cooling techniques

Configuration	Dim silicon mode	
	Frequency (GHz)	Limitation
Air cooling	8×1.6	Temperature < 85 ºC
Liquid cooling	8×2.0	Power<75.4W(0.65V×116A)
Liquid cooling & Static pin switching	8×3.0	Power<111.6W(0.775V×144A)
Liquid cooling & Dynamic pin switching	8×2.0 or 8×3.0	Power < 75.4W or 111.6W

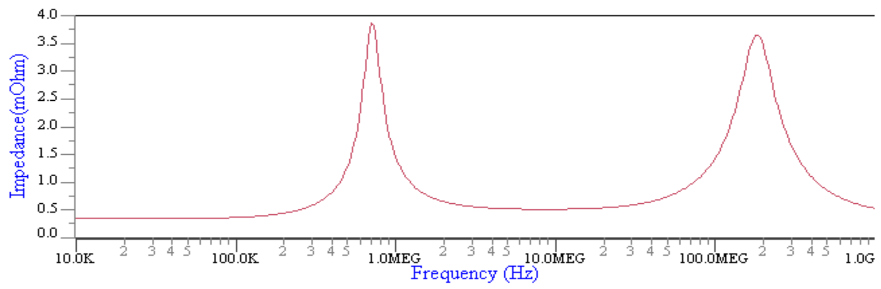
different for different processor frequency as shown in Table 3-2. Figure 3-3 presents the dynamic IR drop while switching from I/O mode to power mode within 0.2μs, 2μs and 20μs. The IR drop fluctuation exceeds 5% for switching time 0.2μs and 2μs, while it is within 5% for 20μs case. Therefore, we use 20μs as the overhead for each pin switching operation. Figure 3-4 plots the impedance for the default I/O mode and the pin switching mode. The impedance does not change much when switchable pins are used for power delivery.

3.2.3 Power Switch

In our design, we use a large power transistor switch with ultra-low on-resistance and low parasitic capacitance. The switch is of comparably large size (like multiple NMOS or PMOS transistors connected in parallel). Figure 3-5 shows a layout design of such a large PMOS transistor switch of W/L=80 based on 16nm technology [67]. Since the estimated resistance of the single switch is nearly 0.47Ω, we connected 262 switches in parallel to achieve the desired 1.8mΩ on-resistance [63] with a 0.232pf parasitic capacitance using 2601μm² of area overhead. Similar calculations for the large NMOS power switch show lower on-resistance and the same parasitic capacitance. The large power switches incur the main processor die area overhead of our design. For 144 switchable pins, they consume 0.00374544cm² of area on the processor die, incurring less



(a)Default I/O mode



(b) Pin switching mode

Figure 3-4. Impedance plots

than a 0.4% area overhead if the total die area is 1 cm².

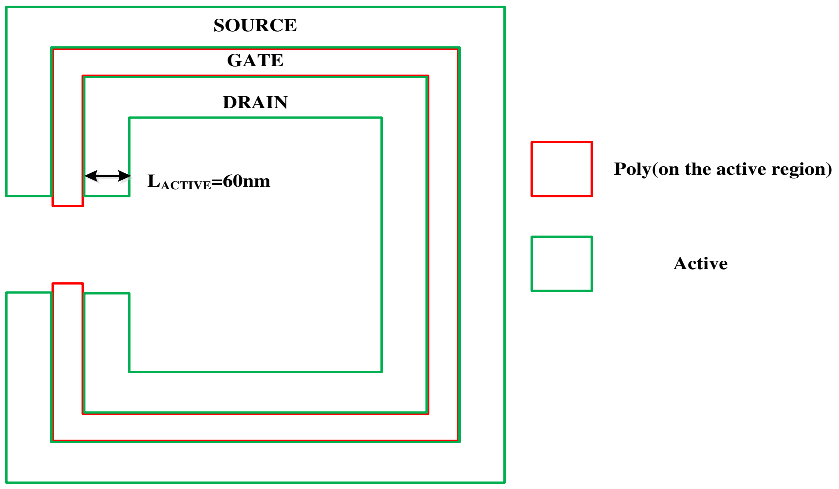
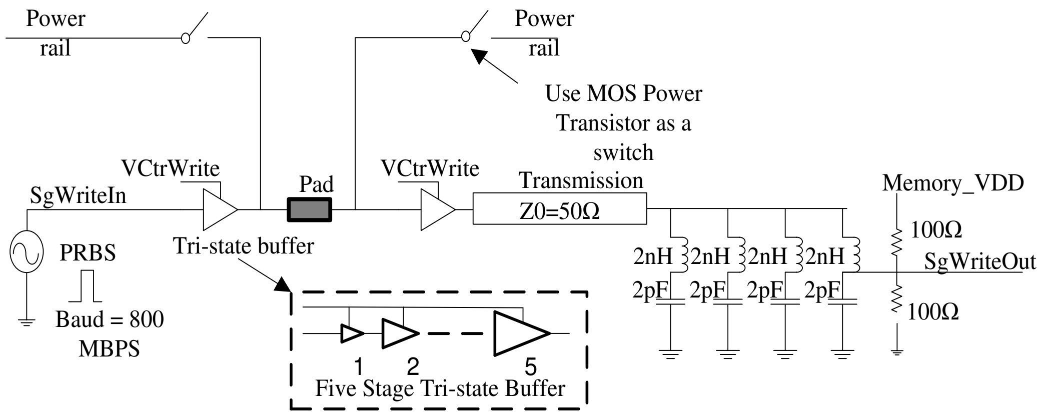


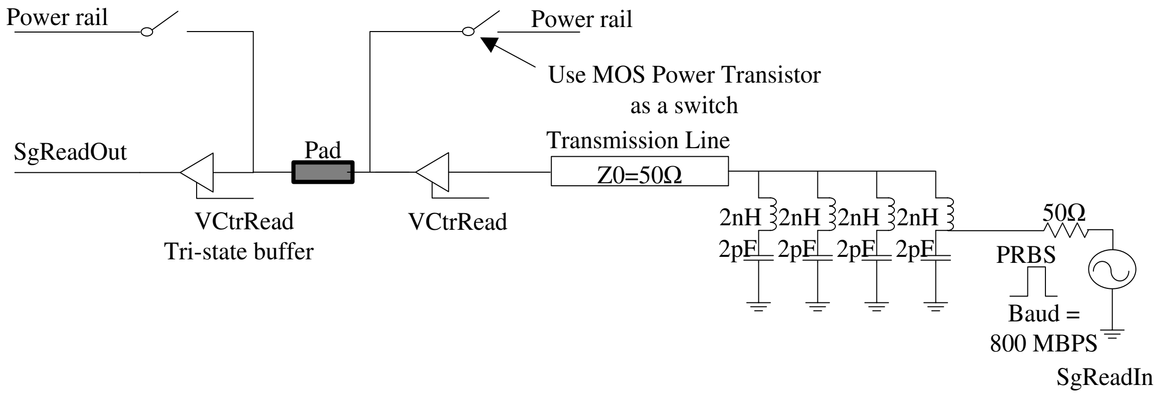
Figure 3-5. Layout of wrapped around large transistor

3.2.4 Signal Transmission

Figure 3-6 shows the circuits of switchable pin design. The switchable pin can either be used for power delivery or the signal transmission. To compensate the parasitic capacitance of the power switches, we add four tri-state buffers for a switchable pin since it can increase signal drive capability. We investigate the impact of adding power switches on the signal transmission path by observing the received eyes for memory

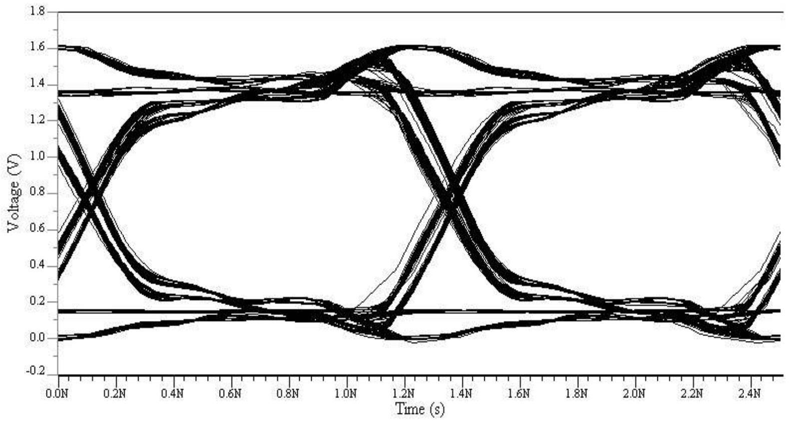


(a)Writing to memory (VCtrWrite=1, the tri-state buffers are enabled while the power switches are off)

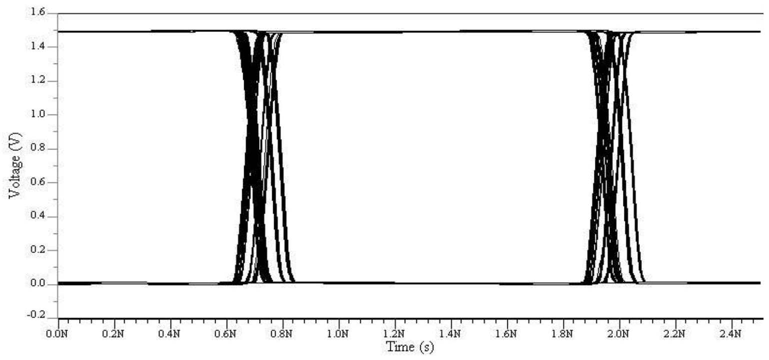


(b)Reading from memory (VCtrRead=1. The tri-state buffers are enabled while the power switches are off)

Figure 3-6. Circuits when a switchable pin is used for signal transmission



(a) Write to memory



(b) Read from memory

Figure 3-7. Received eye diagram

writing and reading. We place both switches and buffers close to the processor, which can minimize the trace shared by power and signal lines. Since buffers on the signal line may cause an impedance mismatch, we have added 50Ω termination impedances on the side of memory devices to match the 50Ω transmission line; these minimize the signal reflections due to impedance mismatching. As shown in Figure 3-7, both eye diagrams show open eyes.

The pin switching design will cause delay on the signal transmission path since extra circuits are introduced as shown in Figure 3-6. For each I/O pin, the extra circuits includes two tri-state buffers and the two shift registers' components.

3.2.5 Thermal Issues

The switchable pins deliver more power in the dim silicon mode as listed in Table 3-2. Our simulation shows air cooling is an unfeasible solution since the worst case processor temperature will be more than 100 ºC in both the dim silicon which will result in serious reliability and lifetime issues. Therefore, we use traditional backplate liquid cooling [33] to increase heat dissipation while delivering more power via dynamic pin switching mechanism.

3.2.6 Dynamic Pin Switching based on Program Phases

Programs tend to show phase behaviors, which can be classified as memory-intensive or computation-intensive. In our design, we will use the switchable pins for off-chip communication to achieve higher communication bandwidth during memory intensive phases. On the other hand, during computation intensive phases where the memory access frequency is low, the switchable pins can be utilized to deliver extra power to mitigate dark silicon. This extra power can either be used to activate dark cores or to increase the frequency of the running processors. Figure 3-8 illustrates the workflow of the pin switching mechanism which favors both the memory intensive and the computation intensive phases dynamically. A predictor, using the program’s history (i.e., patterns of performance counters), is employed to predict the memory usage in the next time interval. When a memory-intensive phase is predicted, the switchable pins will be used for off-chip communication; otherwise, the switchable pins will be utilized to deliver power. Predictions are made in real-time, meaning incorrect predictions can be corrected in the next time interval.

3.2.7 Prediction Model

In this section, we describe the prediction model training procedure employed by the dynamic pin switching scheme. In general, the goal of the predictor is to determine whether the bandwidth requirement of the upcoming intervals is high (or low) enough to require a pin switching for optimal performance. The prediction model is trained as follows.

First, we run workloads on the processor and collect common performance metrics including branch mispredictions and cache misses from all cores and shared components at a preset frequency. By doing this, we obtain the following tuple from each time interval: $< X_1^1, X_1^2, .. X_1^p, X_2^1 \ldots X_2^p \ldots X_q^p, X_S^1, .. X_S^r, MB >$ where each variable X_a^b represents a performance metric of a specific component. The subscript is the component identity (e.g., core ID) and the superscript b corresponds to the index of the metric. For example, X_1^2 denotes the second performance metric observed on the first core. We assume that the number of cores on chip is q and we monitor p performance metrics for each of them. This results in a total of p×q metrics from the integrated cores. The r variables with the subscript S (i.e., X_S^1 through X_S^r) indicate the performance metrics from shared components such as the last-level cache. In this work, we collect 180 counters from each core and 20 counters from the shared components for each time interval. The notation MB represents the average memory bandwidth of this interval.

Second, we reorganize the collected data and train a statistical model to correlate the historical execution behaviors and the memory bandwidth in future intervals. To form a training instance, we combine input variables (i.e., all X_a^b) from M consecutive intervals and use all of them as the input for this sample. The response value (i.e., output) of this training instance is a Boolean flag which is defined as follows. We calculate the average bandwidth of intervals M+1 to M+N; if the average value is greater than a preset threshold, we set the flag to 1, indicating the following N intervals require high memory bandwidth. In contrast, if the average bandwidth is less than the threshold, the flag will be set to 0. By doing this, we are essentially building a rigorous relationship between past execution behaviors (i.e., interval 1 to M) and the future bandwidth requirement (interval M+1 to M+N). After obtaining these training instances, we employ a regression tree model [35] to select 10 input factors that most significantly impact the output value (i.e., the Boolean flag). We then feed the chosen 10 variables, along with their corresponding responses, to a model implementing a bump-hunting algorithm [37] in order to generate a set of rules to guide the pin switching. The rules are interpreted in a group of "IF-ELSE" conditions and are able to identify the regions with the maximum output values. We keep

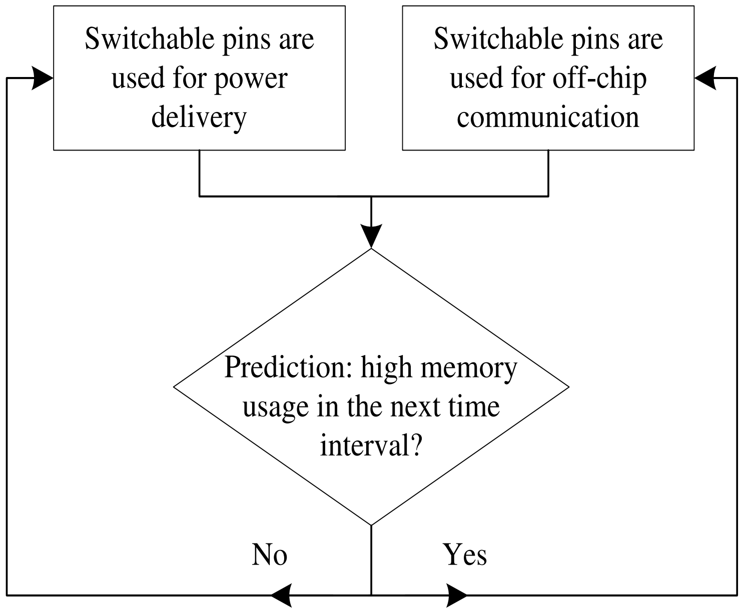

Switchable pins are used for power delivery
Switchable pins are used for off-chip communication
Prediction: high memory usage in the next time interval?
No
Yes

Figure 3-8. Workflow of dynamic switching

comparing the collected performance metrics to the generated rules at runtime. When the conditions are satisfied, a pin switching will be triggered to deliver more power or bandwidth to the processor to improve performance. Note that we randomly sample 80% of all the instances for training and use the remaining 20% for validation as the conventional statistical model training does.

3.3. EXPERIMENTAL SETUP

We simulate an 8-core chip multiprocessor (CMP) and set the maximum allowable temperature to be 85 ºC. Power constraints lead to numerous execution modes in terms of different core frequencies and the number of active cores. For example, decreasing the frequency is effective for reducing per-core power consumption, thus enabling more cores to run simultaneously without exceeding the power limits. For simplicity, we conduct two groups of studies to make our observations and conclusions more comprehensive. The first category of the study is mainly concentrated on the dim silicon mode. We use the term “dim silicon” to refer to the scenarios where all 8 cores are kept active but running at a lower frequency to comply with the power constraints. We explore 13 frequency levels from 1.6GHz to 4.0GHz with a step frequency of 200MHz on the target CMP. Thermal and power constraints cause the core frequency and number of active cores to be different depending on the execution mode. The specific configurations of each execution mode are listed in Table 3-3.

We use use McPAT [56] for processor power modeling with the corresponding parameters listed in Table 3-3. We modify HotSpot [13] to simulate the floorplan shown in Figure 3-9 using air cooling and backplate liquid cooling. SPEC2006 [76] multi-

Table 3-3. Parameters of the performance and power models

Parameters	Values
Technology	16 nm
Die area	10mm × 10mm
Voltage(V)	0.6, 0.625, …, 0.875, 0.9
Frequency (GHz)	1.6, 1.8, …, 3.8, 4.0
Fetch / Issue/ Commit Width	4/ 4/ 5
INT/ FP Window Size	96/ 64
LoadStore/ INT/ FP Units	2/ 2/ 3
Load/ Store Queue Size	80/ 80
Latency of INT ALU/ Mult/ Div	1/ 4/ 12 cycles
Latency of FP ALU/ Mult/ Div	1/ 2/ 10 cycles
L1 Instruction/ Data Cache Size	64/ 64 KB
L1 ICache/DCache Associativity	8/ 8
L1 Instruction/ Data Block Size	64/ 64 B
L2 Cache Size	16 MB
L2 Cache Associativity	16
L2 Cache Block Size	64 B
Memory parameters	
Number of channels	3
Frequency	800MHz
Data bus width	64
Peak memory bandwidth in I/O mode: 38.4GB/s	
Peak memory bandwidth in power mode: 9.6GB/s	

program benchmarks are used in the evaluation of dim silicon mode. We use SimPoint 3.2 [39] to choose a representative block of 200 million consecutive instructions for each SPEC2006 program. Eight copies of the representative instructions are used to create a multi-program workload. The multi-program workloads can be categorized into two types: the first mixes eight copies of the identical SPEC2006 programs, while the second type mixes eight copies of different SPEC2006 programs shown in Table 3-4.

As for the prediction model and online pin switching, we use the execution behaviors in the previous three time intervals to predict the bandwidth in the next interval, with each interval lasting for one millisecond. In this case, the 20μs overhead is 2% of a time interval. Another important parameter is the memory bandwidth threshold, which is

Table 3-4. Simulated multi-program workloads

Name	Combinations
BZIP2	8×BZIP2
MCF	8×MCF
GOBMK	8×GOBMK
DEALII	8×DEALII
HMMER	8×HMMER
SJENG	8×SJENG
LIBQUANT	8× LIBQUANTUM
H264REF	8× H264REF
LBM	8× LBM
P8MIX1	4×NAMD + 4×MCF
P8MIX2	4×NAMD + 4×BZIP2
P8MIX3	4×BZIP2 + 4×SJENG
P8MIX4	2×BZIP2 + 2×DEALII + 1×HMMER + 1×GOBMK + 1×H264REF + 1×SJENG

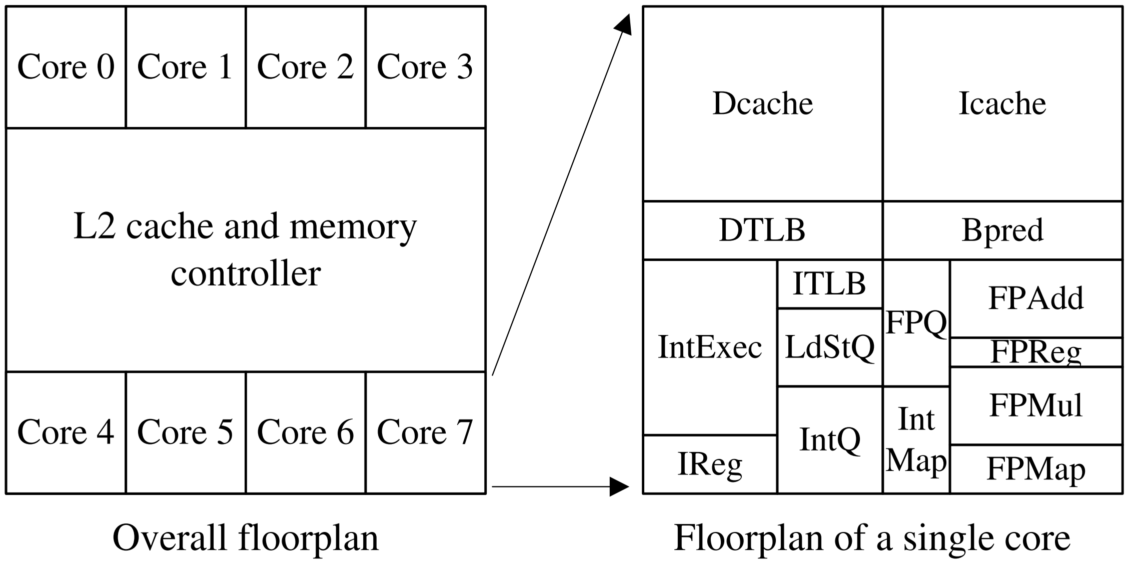


Figure 3-9. Floorplan of the chip multiprocessor

used to evaluate if a pin switching is needed. The threshold should be less than 9.6GB/s, which is the peak memory bandwidth in power mode as listed in Table 3-3. We set the bandwidth threshold to 1.6GB/s in this work to achieve optimal overall performance. Note that these empirically selected parameters do not impact the effectiveness of our proposed scheme and can be changed to other values in a practical system.

3.4. RESULT ANALYSIS

In this section, we demonstrate the effectiveness of the pin switching mechanism by comparing the performance between traditional designs and our proposed scheme.

3.4.1 Rules Explanation

We start by analyzing the generated rule-set used to guide pin switching at runtime. In the dim silicon execution mode, all 8 cores are kept busy and the processor frequency can switch between 2.0GHz and 3.0GHz. Since the frequency is changing, two individual prediction models are necessary to guide the power-to-I/O (i.e., 3.0GHz to 2.0GHz) and the I/O-to-power (i.e., 2.0GHz to 3.0GHz) pin switching. Recall that our pin switching technique is, in essence, a one-way conversion. Therefore, the switch from power to I/O mode means the procedure of returning to the default I/O pin configuration. Assuming that the switchable pins are currently on the power path and the processor is running at 3.0GHz, the following rules indicate that the upcoming interval is very likely to be memory-intensive where off-chip memory access is frequent, and therefore the switchable pins should be switched to the I/O path:

$$\begin{aligned}&\mathrm{Int3\_L2\_readmiss} > 20250 \;\&\&\; \mathrm{Int2\_L2\_readmiss} > 1072\\ &\&\&\; \mathrm{Int1\_L2\_readmiss} > 2293 \;\&\&\; \mathrm{Int3\_L2\_linefill} > 20255\\ &\&\&\; \mathrm{Int3\_L2\_access} > 39942 \qquad 3\text{-}1\end{aligned}$$

The conditions are expressed in a format of IntID_component_metric > X, meaning that the performance counter metric of component in interval IntID (one of the M intervals used as input) should be larger than a certain value X. Given this notion, the first condition in the rule-set listed above indicates that the read misses in the L2 cache in the immediately preceding interval should be larger than 20,250; recall that we use 3 intervals to predict the ensuing interval. Similarly, the second and third conditions set the lower bound for the L2 read misses in the second (Int2_L2_readmiss) and the first previous intervals (Int1_L2_readmiss) respectively. The forth and the fifth conditions set the lower bound for the number of L2 cache line filling and access respectively. It might be followed by memory-intensive execution periods after the intervals with more L2 cache misses, line filling and accesses, so it is reasonable to set the switchable pins for power delivery.

When the swistchable pins have been set for signal transmission and the processor is running at lower frequency, we also need a rule-set to govern when to switch to the power path. The corresponding rules are listed as follows.

$$\text{Int3\_L2\_readmiss} < 20631\ \&\&\ \text{Int3\_L2\_access} < 12032 \qquad \text{3-2}$$

The rules can be explained similarly and we thereby omit the analysis.

3.4.2 Dim Silicon Result

Recall that in the dim silicon mode all 8 cores are enabled while running at a low frequency determined by the power delivery and cooling configurations listed in Table 3-2. Figure 3-10 shows the normalized performance for multi-program under four evaluated configurations. In the air cooling mode, the 8 cores are running at 1.6GHz because the TDP, restricted by thermal constraints, is relatively small. Using liquid cooling, we are able to raise the frequency to 2.0GHz. The remaining two configurations both implement the pin switching mechanism using liquid cooling; therefore there is extra power allowing the core frequency to go up to 3.0GHz. These two final configurations use different pin switching schemes. The first uses a static scheme in which the switchable pins are always set to the power delivery path throughout the entire execution. The second configuration uses a dynamic pin switching scheme guided by the prediction model. Note that all results are normalized against the baseline configuration which uses air cooling.

As shown in Figure 3-10, the scaling trends for most benchmarks are reasonable because higher frequencies lead to faster execution. However, the relative performance among the four configurations is different for different benchmarks. For example, while running 8 copies of MCF and DEALII, the static switch scheme (3.0GHz) has longer execution time compared to the runs with a lower frequency (2.0GHz). Similar trends can also be observed from the execution of P8MIX1, which includes the memory-intensive program MCF. The main reason for the longer execution time here is the substantial penalty from lower memory bandwidth in 3.0GHz compared with 2.0GHz case. More details will be given shortly to expound upon this observation. On the other hand, for applications that are intrinsically computation-intensive, executions using the pin

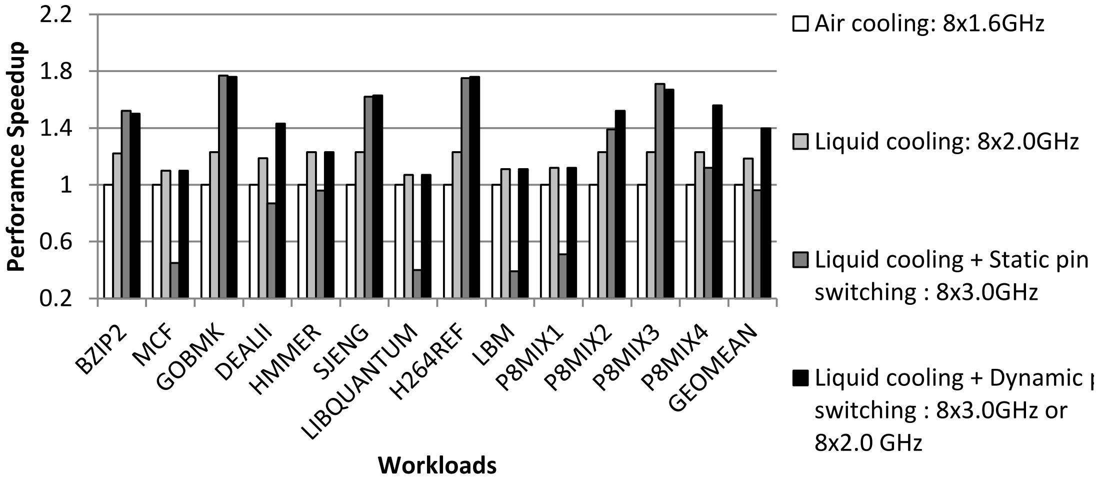


Figure 3-10. Performance speedup when the processor is in dim silicon mode

switching technique will significantly outperform those with traditional configurations. For these benchmarks, even the static pin switching leads to an impressive speedup because memory-bound intervals are fairly rare during the execution. Therefore, maintaining a higher core frequency is more beneficial.

Furthermore, in benchmark DEALII when the pin switching is guided by the prediction model, we notice further performance enhancement compared with the static switching. This is because with the dynamic approach, the predictor will estimate how much off-chip traffic will be generated during upcoming execution period, thus determining the most appropriate path for the switchable pins. Compared with the static scheme which blindly sets the switchable pins to the power delivery path, the dynamic switching strategy can more effectively balance the requirement of power delivery and off-chip bandwidth. In general, the geometric mean of the performance speedup delivered by our optimal scheme (liquid cooling + dynamic pin switching) is 1.39X compared with the baseline (air cooling).

To further understand the scaling trend of each workload, we plot the number of L2 cache misses per 1K instructions in Figure 3-11. The figure shows whether a workload is computation-intensive or memory-intensive. In addition, a high-accuracy predictor stands as one of the most important factors in determining the effectiveness of dynamic switching; therefore it is necessary to evaluate the accuracy of our prediction model. Recall that, in our model, the response of each training instance is set as a Boolean flag. Consequently, by counting the occurrences of true positive (TP), true negative (TN), false positive (FP), and false negative (FN), we calculate the prediction accuracy as follows:

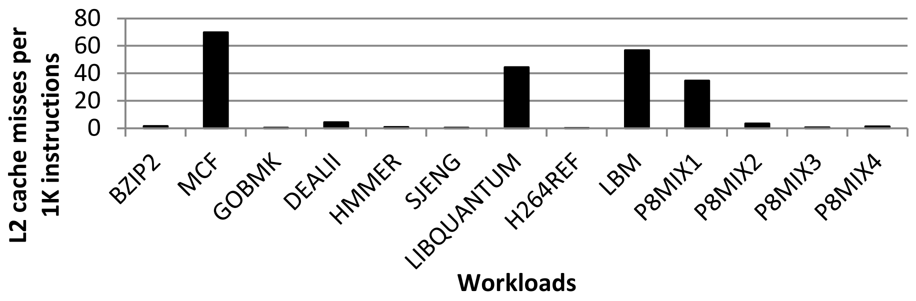


Figure 3-11. Number of L2 cache misses per 1K instructions on a processor configured to 8×2.0GHz (liquid coiling)

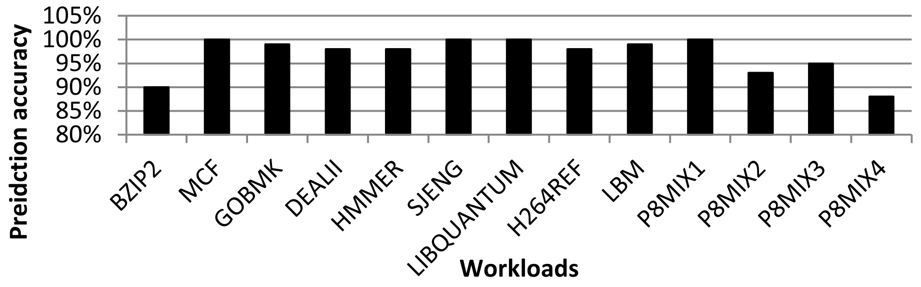


Figure 3-12. Prediction accuracy on a processor in dim silicon mode

$$\text{Accuracy} = \frac{\text{TN+TP}}{\text{TN+TP+FN+FP}} \qquad 3\text{-}3$$

As shown in Figure 3-12, the prediction accuracy is fairly high for most benchmarks. For applications where the accuracies are slightly lower, the predictor still results in impressive performance improvements over the static switching scheme.

3.5. RELATED WORK

Dark silicon: Dark silicon has emerged as an increasingly important issue that will menace the scaling of Moore's Law in the deep submicron era and beyond. Esmaeilzadeh et al. [36] use an analytical model to predict processor scaling for the next few generations. They demonstrate that dark silicon will be heavily exacerbated by the continued shrinking of manufactured technology. Researchers [36][38][40][41] commonly attribute the cause of dark silicon to physical power and off-chip bandwidth constraints. Kim et al. [49] proposes to integrate memory with processors as a 3D chip. This integration can mitigate off-chip bandwidth constraints but it brings more challenges for power delivery and cooling since extra power will be consumed by the integrated memory. Hardavellas et al. [41] investigate this problem and believe even if an advanced liquid cooling technique was applied the power delivery would still result in dark silicon.

Shortage of C4 pads: The ITRS [15] predicts that C4 pad density will increase 7.7% annually, and fail to ever meet demand, which is increasing at 15.7% annually. Zhang et al. [90] evaluate the usage of C4 pads in a multicore processor and conclude that we will see a C4 pad shortage starting from 16nm technology node. The shortage comes from an increasing demand in power delivery and off-chip bandwidth but a slow improvement in C4 pad technology. Previous works [40][73] observe that the required number of C4 pads increases exponentially with the number of processor cores. Therefore, an exponentially larger number of C4 pads are needed to increase off-chip communication. Moreover, more power pads are needed for current delivery as each new technology increases the power density. Researchers [21] from IBM also observe the C4 pad shortage and propose to utilize the heat sink to deliver power. A recent work demonstrates the impact of the pad shortage on power delivery quality [91]. In another recent work, Chen et. al [28] propose to use switchable pins to increase memory bandwidth. Instead, we propose to increase power delivery using pin switching.

On-chip voltage regulator (VR): Theoretically, an on-chip VR can be used to deliver more power by supplying a chip with a relatively high voltage and convert the voltage to a normal value inside the chip. However, on-chip VR has large area [51]. Our proposal presents another alternative approach to the power delivery problem.

CHAPTER 4. BOOSTING OFF-CHIP BANDWIDTH WITH PCM VIA SWITCHABLE PINS

4.1. BACKGROUND

The scaling of memory technology has improved memory subsystems with increasing density, growing capacity and decreasing cost over the past decade. However, this scaling faces challenges since the shrinking size of cell leads to a smaller capacity for storing charges. This trend increases leakage power and refresh-rate frequency, and thus reduces energy efficiency and bandwidth of memory devices. Given these challenges, scaling DRAM beyond 40 nanometers will be increasingly difficult [75]. Phase-change memory (PCM) is a promising candidate to replace conventional memory technology to enable the continuous scaling of memory technology [60].

There are several memory subsystems proposed by architect to replace conventional memory devices using PCM devices [54][64][89]. We evaluate the benefits of switchable pins based on the performance of a PCM subsystem [60]. Though PCM has recently seen continuously decreasing access latency, it is still several times larger than that of DRAM. Pin Switching increases off-chip bandwidth, and also reduces this memory subsystem access latency. Thus, it may alleviate the drawbacks of PCM by reducing the queuing delay of memory requests. Furthermore, PCM has relatively longer write latency and thus reduces the utilization of off-chip bandwidth since a write will hold the entire bus until it is completed. Pin Switching mitigates this problem by allowing more simultaneous in-flight memory requests. In the section, we also include the performance of multi-thread workloads for switchable pins.

4.2. EXPERIMENTAL SETUP

To evaluate the benefit of our design, we use the identical configuration for simulated system shown in Table 2-4 and selected workloads shown in Table 2-5. Additionally, we employ a timing model of PCM based on [60] and four multi-threaded workloads including art [78], lbm [60], srad [27] and backprop [27] to evaluate the performance of Dynamic Switching shown in Table 4-1. We manually select memory-intensive regions from the workloads and run 100 million instructions per thread in each workload. The regions are independently executed to gather instructions per cycle (IPC), last-level-cache misses per 1,000 instructions (LLC MPKI), row buffer hit ratio, and the bandwidth displayed in Table 4-1. The bandwidth and LLC MPKI numerically portray the memory access intensity, making them indicators of our design's potential benefit. Row buffer hit ratio reveals the memory access locality and latency. Programs with low

Table 4-1. Benchmark memory statistics

Benchmark	IPC	LLC MPKI	Row buffer hit ratio	Bandwidth(MByte/s)
libquantum	0.30	58.14	96%	4441.57
milc	0.16	41.86	81%	3641.48
leslie3d	0.62	20.72	85%	3311.84
soplex	0.31	31.34	80%	2501.53
lbm	0.36	23.12	87%	2151.90
mcf	0.15	57.54	19%	2138.81
astar	0.25	29.12	51%	1871.53
omnetpp	1.38	0.49	83%	172.09
gromacs	1.34	0.38	82%	129.60
h264	1.13	0.13	32%	38.03
bzip2	1.13	0.12	94%	35.54
hmmer	1.95	0.00	38%	0.28
art (OMP)	1.4	17.56	88%	6390.85
lbm (OMP)	2.72	8.24	57%	4862.41
srad (OMP)	2.16	12.04	62%	6838.84
backprop (OMP)	0.4	69.61	94%	7203.58

row buffer hit ratios suffer from longer bank access latency due to the row buffer miss penalty. Longer memory accesses increase the queue delay which impedes the fol-lowing incoming requests in the buffer.

4.3. RESULTS

4.3.1 Memory-Intensive Multi-threaded Workloads

Figure 4-1 and Figure 4-2 show that Dynamic Switching with classic stride prefetching improves performance and increases consumed off-chip bandwidth of multi-threaded programs. All results are normalized against the baseline. Dynamic Switching and the stride prefetching with the degree 4 improve performance by an extra 102% in geometric mean providing the best performance compared to the baseline. Prefetching can exploit the benefits of multi-bus mode for multi-threaded programs, increasing the consumed off-chip bandwidth shown in Figure 4-2. For instance, Dynamic Switching and the prefetching with degree 4 yields an extra 29% performance improvements compared to the baseline with the same prefetching degree for the art workload, while Dynamic

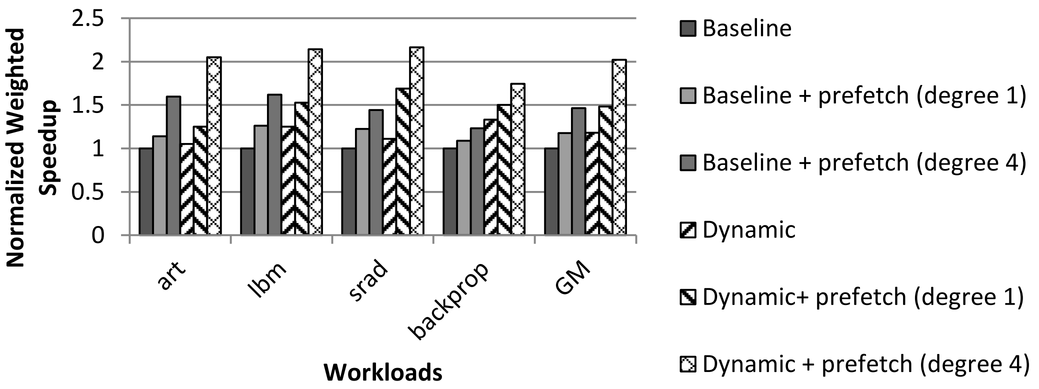


Figure 4-1. Performance evaluation of multi-threaded workloads with Dynamic Switching and prefetching (degree = 1, 4).

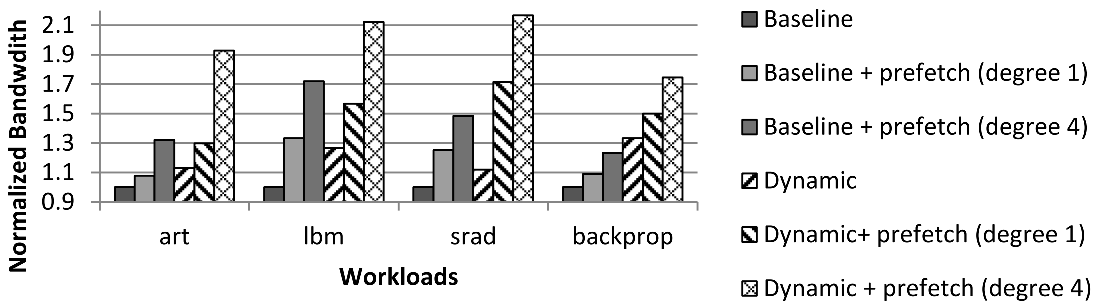


Figure 4-2. Normalized consumption of off-chip bandwidth of multi-threaded workloads using Dynamic Switching and prefetching (degree = 1, 4)

Switching delivers a mere 5% performance improvement compared to the baseline without prefetching. We conclude that this benchmark cannot generate am adequate number of memory requests to saturate the off-chip bandwidth, and thus benefits from the prefetching which can increase memory level parallelism.

4.3.2 Memory-Intensive Multi-programmed Workloads using PCM

Figure 4-3 shows the performance improvement of Dynamic Switching combined with a stride prefetcher (degree = 1,2,4) for memory-intensive workloads running on the PCM subsystem. The results are normalized against the weighted speedup of Dynamic Switching without a prefetcher. Dynamic Switching consistently delivers performance benefits for all workloads and achieves an average weighted speedup of 1.97 in geometric mean without prefetching. Dynamic Switching and the stride prefetcher (degree 4) achieve the largest performance improvement with an average weighted speedup of 2.27. The prefetcher yields an extra 0.54 weighted speedup compared to Dynamic Switching and the baseline using a stride prefetcher (degree=4). The performance improvement stems from increasing off-chip bandwidth as shown in Figure 4-4. Dynamic switching without a prefetcher increases the off-chip bandwidth by 58% compared to the baseline,

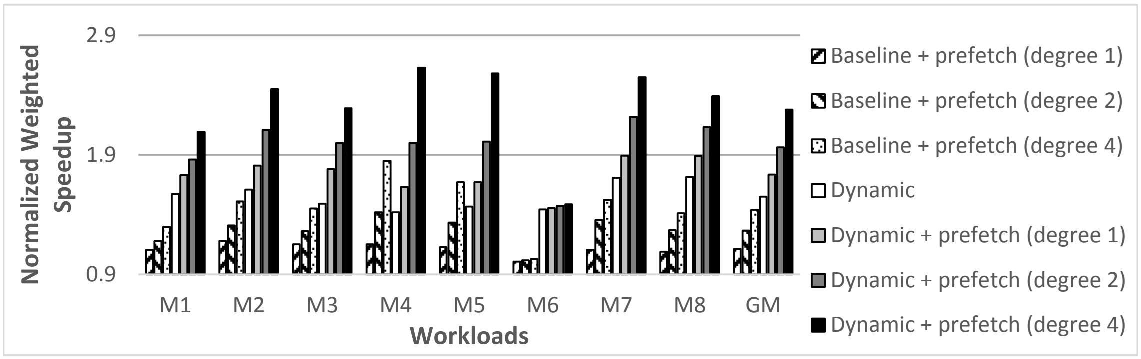


Figure 4-3. Improved throughput of Dynamic Switching boosted by stride prefetchers (degree = 1, 2, 4) for memory-Intensive workloads using PCM

while the prefetcher (degree 4) increases off-chip bandwidth by 22% in comparison to Dynamic Switching and the baseline. Dynamic Switching and the prefetcher exhibit remarkable performance improvements and increase off-chip bandwidth for all workloads except M6. Dynamic Switching still deliveries considerable performance benefits for M6 though the prefetcher delivers little benefit as M6 suffers from the low latency of row buffer misses and has irregular access patterns which are hardly captured by the stride prefetcher.

4.3.3 Memory-Intensive Multi-threaded Workloads using PCM

Figure 4-5 and Figure 4-6 show that Dynamic Switching with classic stride prefetching improves performance and increases consumed off-chip bandwidth of multi-threaded workloads on a PCM subsystem. All results are normalized against the baseline. Dynamic Switching and the stride prefetching with degree 4 improve performance by an extra 130% in geometric mean providing the best performance compared to the baseline. Prefetching can exploit the benefits of Pin Switching, increasing the consumed off-chip bandwidth as shown in Figure 4-6. Additionally, Dynamic Switching can mitigate the performance loss caused by the longer latency of row buffer misses in PCM. For instance,

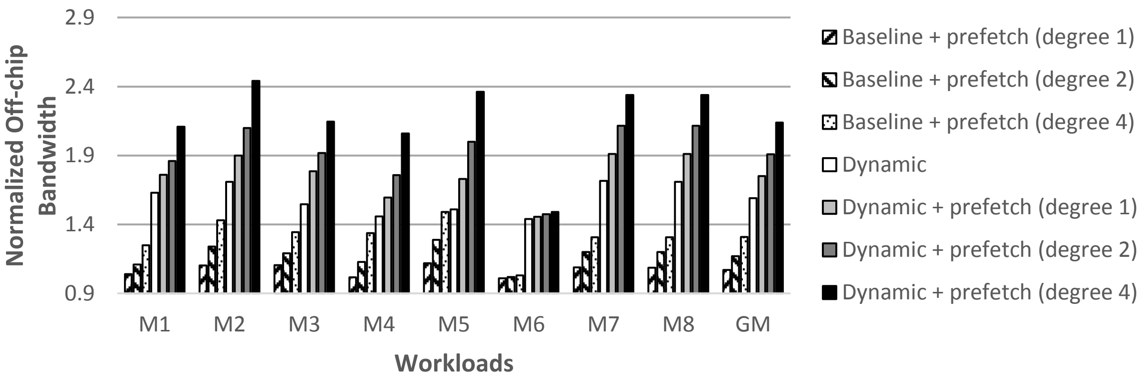


Figure 4-4. Normalized off-chip bandwidth of Dynamic Switching boosted by stride prefetchers (degree = 1, 2, 4) for memory-Intensive workloads using PCM

`

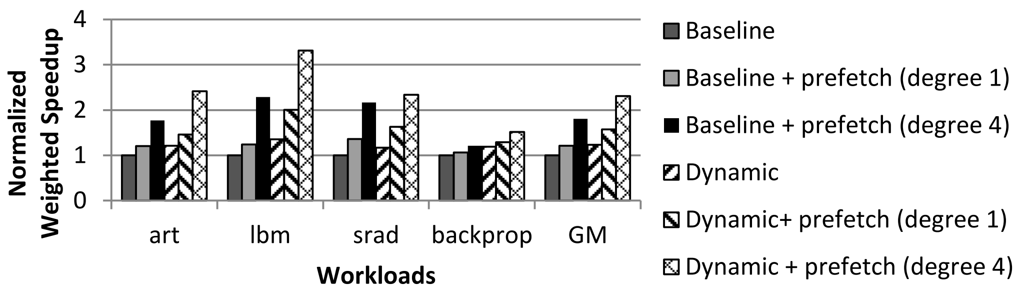


Figure 4-5. Performance evaluation of multi-threaded workloads using Dynamic Switching and prefetching (degree=1, 4) on the PCM subsystem

it achieves the highest performance improvement in the lbm workload which has the lowest row buffer hit rate of 57%.

4.3.4 Mixed Multi-program Workloads on the memory subsystem using PCM

Figure 4-7 shows the performance improvement of Dynamic Switching combined with a stride prefetcher (degree = 1, 4) for mixed workloads running on the PCM subsystem. The results are normalized against the baseline. Dynamic Switching with prefetching yields considerable performance benefits for all the mixed workloads and achieves an average weighted speedup of 1.26 in geometric mean. The combination of Dynamic Switching and the prefetching with a degree of 4 yields slightly more

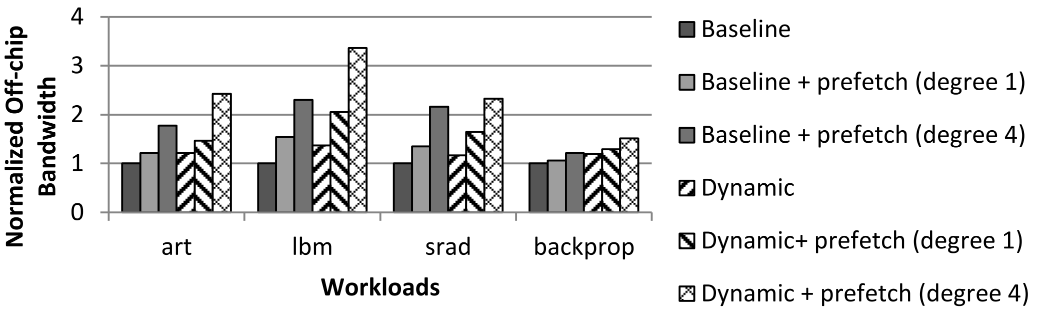


Figure 4-6. Normalized consumed off-chip bandwidth of multi-threaded workloads using Dynamic Switching and prefetching (degree =1, 4) on the PCM subsystem

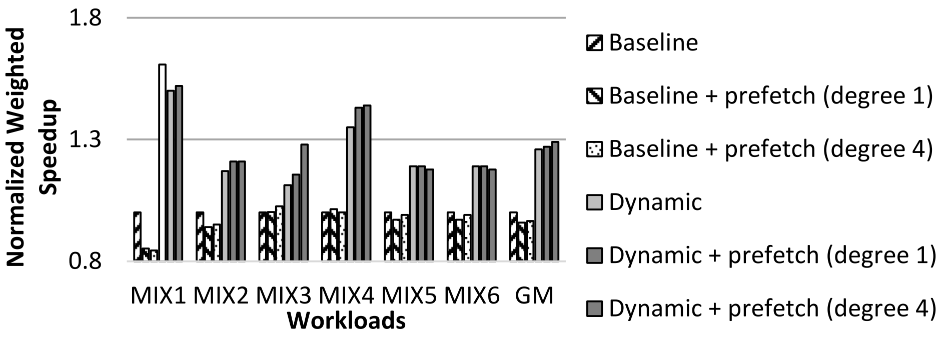


Figure 4-7. The improved throughput of Dynamic Switching boosted by stride prefetchers (degree = 1, 4) for mixed workloads with PCM

performance improvements than Dynamic Switching without prefetching, and delivers an average weighted speedup of 1.29, while the baseline using the same prefetching decreases the average performance by 4%. Prefetcher might increase the latencies of off-chip memory requests from the cores by generating additional requests which compete for the already insufficient off-chip bandwidth.

4.4. CONCLUSION

Limited off-chip memory bandwidth has been widely acknowledged as a major constraint preventing us from obtaining commensurate performance benefits from the faster processor cores. This is especially challenging in the current multi-core era due to a high volume of memory requests coming from an increasing number of processor cores. To alleviate the shortage of off-chip bandwidth, we propose an innovative pin switching technique which dynamically allocates pins for power delivery or signal transmission with minimal changes to the circuit. By accurately identifying memory-intensive phases at runtime, the proposed strategy converts a portion of the pins used for power delivery to signal transmission mode, providing additional off-chip bandwidth and improving the overall performance. As shown by the evaluation results, along with other techniques including Dynamic Switching and stride prefetching, our scheme is capable of significantly accelerating the program execution for both multi-programmed and multi-threaded workloads. Our evaluation also shows that Dynamic Switching can improve the performance of PCM subsystems.

CHAPTER 5. INCREASING INTER-SOCKET BANDWIDTH VIA SWITCHABLE PINS

5.1. RELATED WORK

Many works are proposed such as increasing the throughput of main memory and the bandwidth of inter-socket communication, since the long latencies of off-chip accesses has been identified as one of the bottlenecks for massive parallel workloads.

Researchers try to boost the throughput of main memory by modifying memory devices, the memory channels, and processors. For DRAM devices, row buffer misses reduce the utilization of the bandwidth since the program will incur a considerable overhead for turning on/off a row. The row buffer is proposed to break the inside of a bank into multiple sub-arrays and thereby reduce the row buffer miss rate, and have a lower overhead for switching the sub-array instead of a whole row [47]. An asymmetric DRAM bank organization is proposed to improve the system performance via using larger rows for system throughput and smaller rows for lower overheads for turning on/off a row [80].

Several works improve the performance of main memory at the rank level. A conventional rank is broke down into mini-ranks that have a shorter data width, and can be operated individually for higher memory system throughput [92]. Increasing the bus frequency is proposed to improve the performance of memory channels via buffering data and commands in the DIMMs [1]. Splitting the data bus into several small buses is also proposed to boost the throughput of memory channels since each small data bus can work independently [86]. Dynamically increasing the bandwidth of the main memory is proposed but it has considerable parasitic capacity from power switches [28]. Our design only switches signal pins and does not have this issue.

From the processor side, works are proposed to improve the performance via scheduling off-chip requests and using DRAM cache. A memory scheduler is proposed to boost system performance based on reinforcement learning that can understand program behaviors [43]. Another memory scheduler is designed to boost multi-threaded performance by providing fair off-chip access of off chip for each thread [65][66]. DRAM cache is proposed to reduce the number of off-chip accesses since it has superior bandwidth than main memory and larger size than SRAM-based cache [70]. Lowering the off-chip traffic and reducing the tag lookup latency further improve the performance of DRAM cache [46]. The works reduce the off-chip traffic between processors and main memory but do not affect the inter-socket traffic.

The bandwidth of inter-socket communication: Silicon photonics have been studied for a long time as a promising technology to replace the electrical off-chip buses and provide superior bandwidth with very low energy consumption [70]. It can boost the bandwidth of main memory while requires re-architecting DRAM memory systems, and increase the bandwidth of interconnect [53]. The photonics interconnect has been developed [82], but is not widely used due to the two factors: the manufacture cost and the reliability issue [77]. The electrical chip-to-chip cost is 0.25$/Gbit, while the current parallel optic transceiver manufacturers state that perhaps $4/Gbit is achievable today. The reliability of silicon photonics interconnects is unclear since the integration of photonic emitters and receivers into the IC may cause some reliability issues. Our design is a cost-effective and reliable solution for inter-socket traffic since it is based on conventional electrical interconnects.

5.2. DESIGN OVERVIEW

We introduce two modes for a multi-socket system: the single-link mode in which the system has default bandwidth of off-chip memory and bandwidth of inter-socket communication, and the multi-link mode in which the system has multiplied bandwidth of inter-socket communication at the cost of lower off-chip memory bandwidth. The two modes are shown in Figure 5-1 as an example in which the system has two processors connected via a QPI bus with 20 lanes and the each processor has four memory channels. This example represents the typical case used in the following discussion easily extended for different system configurations. In the example, the multi-link mode multiplies the bandwidth of QPI by a factor of 3 and loses two memory channels since the number of pins for a memory channels is more than the number of pins for a QPI bus. This

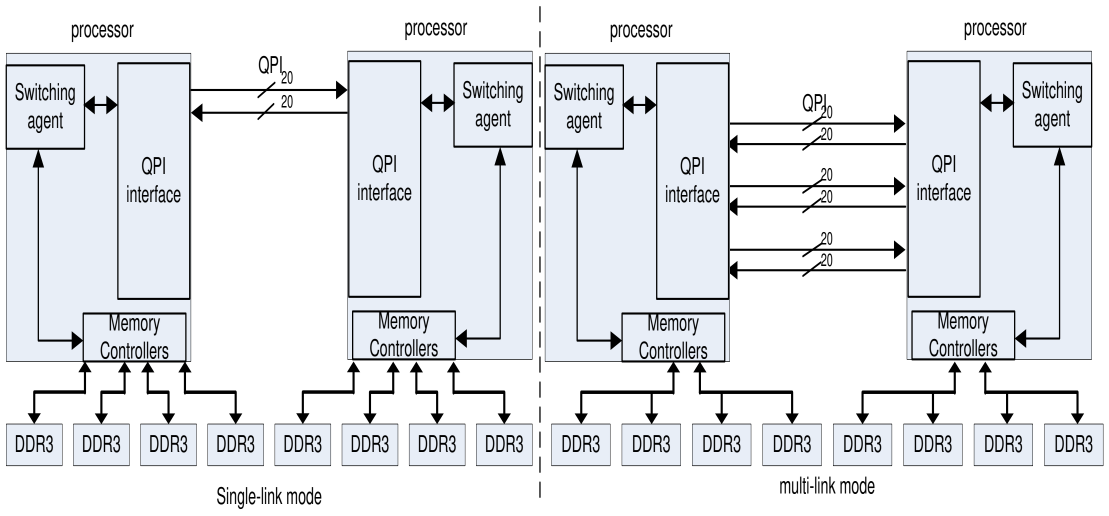


Figure 5-1. The simulated system running in the single-link mode and the multi-link mode

calculation is based on the fact that a memory channel requires 125 pins from a processor to access memory devices [1], while a QPI bus demands 84 pins from processors [17].

The design needs a hardware unit to orchestrate the switch to quickly capture the phase of intensive inter-socket communication since intensive phases could be abrupt and short. The design introduces a switching agent for each socket to coordinate increasing the inter-socket communication and decreasing the off-chip memory bandwidth. The agent switches the function of switchable pins from accessing off-chip memory to communicating between sockets via signal switches siting on the die and the motherboard. It also controls the DRAM controllers to adapt the less off-chip bandwidth and the Quick Path Interconnect (QPI) to utilize the extra bandwidth of inter-socket communication. The switching agents from all the processors have to reach an agreement that the system can increase its throughput via a switch rather than a subset of processors. With a bottom-up approach, we discuss the mechanism of switching off-chip bus connection as well as auxiliary circuits in the chapter 5.2.1, the modification of DRAM controller and QPI physical layer is address in the chapter 5.2.2 and the chapter 5.2.3, and switching agents and the switching conditions in the chapter 5.2.4 and the chapter 5.2.5.

5.2.1 Off-chip connection

The modified off-chip connection in the two modes is shown in Figure 5-2 with an auxiliary circuit named as signal switch. We only show the related off-chip bus connection for a processor with a pair of QPI data lane, since the simulated system is homogenous and the off-chip memory buses per socket are identical to each other. The auxiliary circuits add more QPI buses on the motherboard from memory buses, while the processors still can read data or write from the memory devices attached to the memory buses in the multi-link mode. The memory devices are attached to another memory buses as an extra rank in the multi-link mode, while are accessed via a dedicated buses in the single-link mode. It maintains the accessibility of data stored in the memory devices though it incurs the extra auxiliary circuits on the motherboards.

A signal switch is employed to switch the function of a pin between accessing off-chip memory devices and inter-socket communication, or to attach two memory channels to one another for increased data accessibility. The signal switch is a classic switch consisting of an n-type metal-oxide semiconductor (NMOS) and a p-type metal-oxide semiconductor (PMOS) each having a relatively low parasitic capacitance and propagation delay. This switch is ideal for high-speed signals that are sensitive to parasitic capacitance and signal delay.

With the signal switches, we can increase the bandwidth of inter-socket communication via switching the system from the single-link mode to the multi-link mode. In the single-link mode, pairs of signal switches (1) on the die connect pins to the memory controllers, and to a dedicated memory channel via pairs of signal switches (2) on the motherboard. The signal switches (3) dis-attach the memory channel from another one and the system has four memory channels. The processor can writes/read data to/off memory devices via the memory channels by turning on the signal switches in the corresponding direction. In the multi-link mode, the signal switches (1) (2) connect the pins to the QPI buses instead of the memory channels, while the signal switches (3) attach the memory channel to another one and the system has two memory channels. The processor can access the memory devices via the two memory channels by turning on the

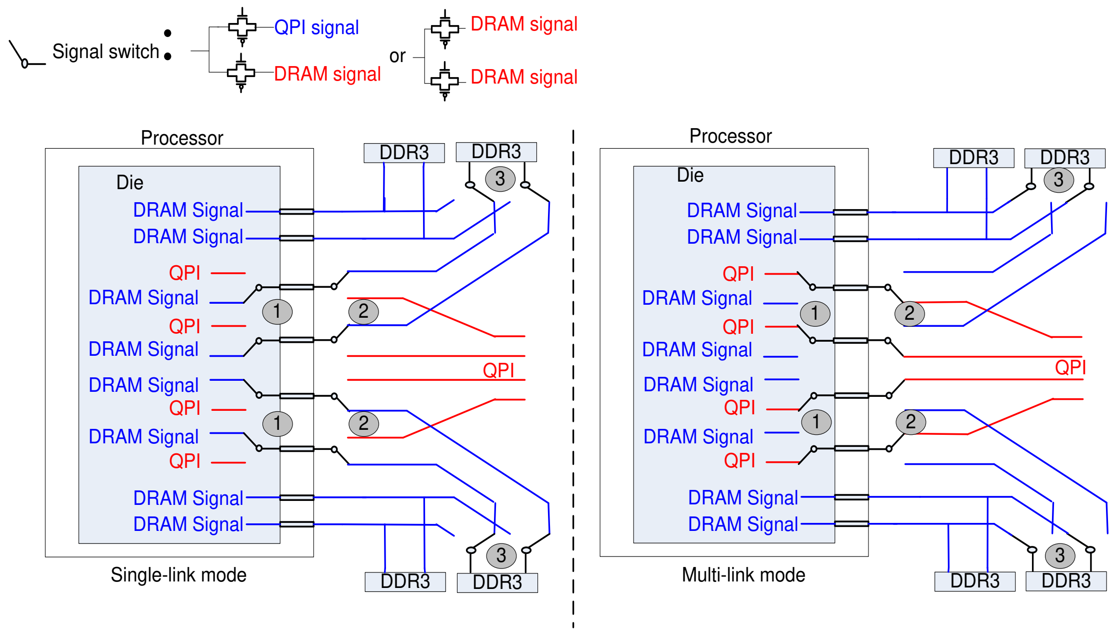


Figure 5-2. The off-chip bus connection in the single-link mode and the multi-link mode

signal switches (3) in the corresponding direction. The location of switches (2) and switches (3) on the motherboards are also vital to the signal integrity. The switches (2) should be placed close to the processors to reduce the signal reflection between the switches (1) (2), while the switches (3) should be placed close to the DRAM devices for the same reason.

5.2.2 Memory controllers

We modified the memory controllers to dynamical change the number of memory channels when the system switches between the single-link mode and the multi-link mode shown in the Figure 5-3. We turn off/on two memory controllers when the system switches from multi-link mode to single-link mode or vice versa. The other two memory controllers handle all memory requests in single-link mode. Given a fixed address mapping policy, this incurs a negligible area overhead to dispatch memory requests to the

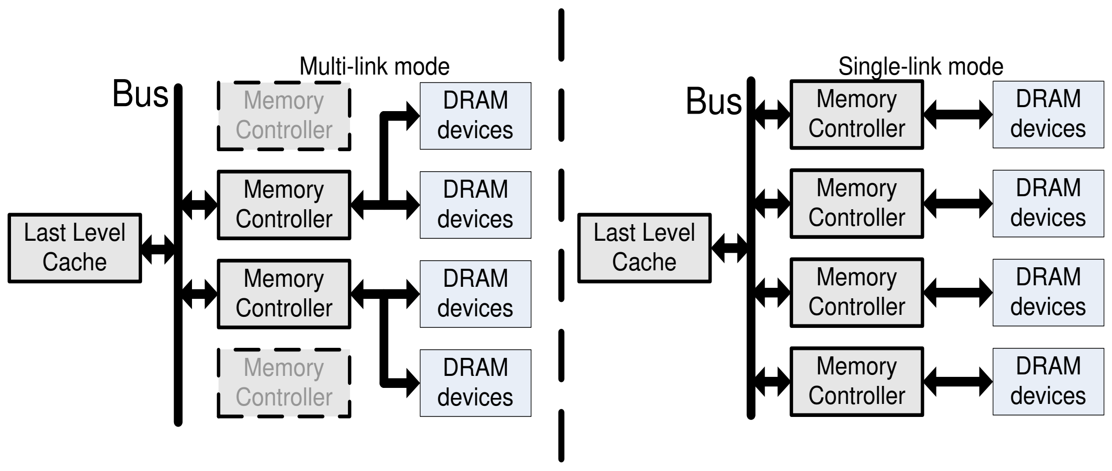


Figure 5-3. The memory controller running in the single-link mode and the multi-link mode

corresponding memory channels, and few extra pins to select the memory channel in single-link mode. The main challenge is that all memory requests have to be committed in memory controllers to switch the system between modes instead of migrating requests cross memory controllers. This overhead is discussed in the runtime overhead section.

The length of write and read request queues is halved when the system switches into multi-bus mode. This can potentially reduce the off-chip bandwidth for main memory. The slowdown is minimal due to low main memory access traffic in multi-bus mode. Additionally, we do consider the slowdown in our simulation.

Consolidating many memory channels could lead a channel to have too many ranks that exceed the standard, which may hurt the scalability of the design. The high speed of memory buses limits the maximal number of ranks in a memory channel. This constraint can be relaxed by lowering the frequency of the memory bus in multi-bus mode. This overhead could be negligible because traffic between the processors and main memory is low in multi-bus mode.

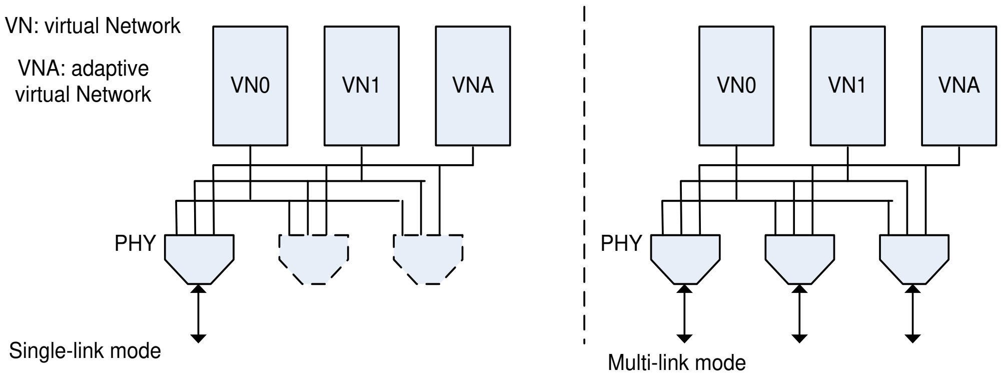


Figure 5-4. The physical layers of QPI running in the single-link mode and the multi-link mode

5.2.3 QPI stack

QPI is a point-point processor interconnects with five layers, physical layer, link layer, routing layer, transport layer, and protocol layer [20]. Each layers works independently with other layers and we only discuss the physical layer and link layers that are related to our modification. We add the extra physical layers (PHY) that are fully connected with virtual networks in link layers to support more than one QPI bus shown in Figure 5-4. The PHYs are powered off in the single-link mode, while they in the multi-link mode can receive packages from other processors or send packages waiting in the buffers of virtual networks. The PHYs are bufferless and thereby can be quickly turned on /off since the link layers control the traffic via credit/debit flow control. We employ switching agents to guarantee that there is no dropping package during the transitions between the multi-link mode and the single-link mode. The switching agents enforce the PHYs can only send packages after the PHYs in the receiver side can accept the packages, when the system switches to the multi-link mode. The switching agents also enforce the senders of the PHYs are turned off before the corresponding receivers of PHY are disabled, when the system switch to the single-link mode.

5.2.4 Switch agents

To switch between the single-link mode and the multi-link mode, we employ a switching agent inside each processor to coordinate the transitions in the two processors. The switching agents analyze the traffic of local off-chip memory access and that of inter-socket communication via collecting hardware counters from the local memory controllers and the QPI controller, which consists of a sender and receiver. Based on this information, the switching agents take the following steps for a transition once they detect a phase in which the performance can be improved by switching the system to multi-link mode:

1. A switching agent called the launcher detects the current phase and sends switching inquiries to other switching agents called assistants via the QPI buses. An assistant denies the switching inquiry by sending a disapproving response to the launcher if it does not detect this phase locally. Otherwise, the assistant accepts the inquiry by sending back an acknowledging response.
2. If the launcher receives a disapproving response, it immediately aborts this transition. Otherwise after it receives all acknowledging response, it turns off the two memory controllers, switches the off-chip connections, and turns on all the extra QPI receivers, while it initializes a transition by sending switching requests to all the assistants that also do the same thing locally once they receive the request.
3. After the switching is done, each switching agent sends responses to its neighboring switching agents.

4. After receiving a response from a neighbor, the switching agent turns on the QPI sender connecting to the neighbor. After all the extra QPI buses are connected, the transition is done.

For switching the system to single-link mode, the switching agents take similar steps for the transition:

1. The launcher detects the phase and sends switching inquiries to other switching assistants via the QPI buses. The launcher will abort the transition if any switching assistants send back a disapproving response to the launcher.
2. After receiving acknowledging responses from all assistants, the launcher disables the extra QPI senders of all neighbors, while it sends switching requests to all assistants that take the same action. Each switching agent sends responses to its neighbors after the action is completed.
3. Once having received the response, a switching agent disables the corresponding QPI receivers. After it has received the responses from all neighbors, it switches the off-chip bus connection and then turns on the two memory controllers.
4. After every switching agent has taken this action, the transition is done.

The launcher in the process can be pre-selected based on the processor ID. We ignore the overhead for leader election, since it is only needed once when all processors are powered on. Note that it is possible to switch a subset of processors into multi-link mode while keeping others in single-link mode. A hybrid approach is not considered in this work for to two reasons: 1. most threads in workloads show similar phases and thereby most processors are likely to benefit from the same mode thus the benefit of a partial transition is minimal compared to that of a full transition. 2. Partial transitions require more complicated synchronizations to find the optimal configuration. Therefore, we do not include the discussion of switching a subset of processors.

5.2.5 Switch condition

Concerning the runtime overhead, we take 0.1ms as the minimum interval for a switching. The launcher will collect the number of un-core requests hitting locally and the number of un-core request hitting remotely. At the end of an interval, the launcher will initialize a switch from single-link mode to multi-link mode if it observes that remote traffic is heavier than local traffic; or a switch from multi to single-link mode if it observes that local traffic is more intensive than remote traffic. To evaluate the performance of this dynamic switching, we introduce the baseline in which the system remains is “unswitched” in single-link mode; static switching in which the system is permanently “switched” via multi-link mode; and dynamic switching in which the system can dynamically switch between single-link and multi-link mode.

5.2.6 Area Overhead & Propagation Delay

The area overheads of the design comes from the signal switches on the die, the modifications of QPI and the switching agent. The area of signal switches, consisting of a pair of large NMOS and PMOS, is negligible since the area of a signal switches is less than the area of 4000 transistors based on 45nm technology. The extra QPI physical layers are buffer-less and incur trivial area overhead. The switch agent for each processor also incurs a negligible area overhead since it uses a straightforward rule and only a few steps to coordinate the switches cross-processors, which are easy to implement in hardware.

The propagation delay caused by signal switches depends on the resistance and capacity of the load. We measure the propagation delays of five cases in Figure 5-5. The Spice models for QPI buses and memory buses in single-link mode and the multi-link mode by comparing the propagation delays with signal switches to the delays without them based on 45nm technology using mentor graphic tools [11]. The longest propagation delays of the QPI bus and memory bus are 0.13ns and 0.12ns respectively. The delay can be further reduced with a better technology.

5.2.7 Runtime overhead

We break down the runtime overhead of the transitions between the two modes into two parts: the runtime overhead of turning on/off memory buses and the runtime overhead of turning on/off QPI buses. The former mainly comes from re-stabilizing the signals on the memory buses and turning on/off the memory controllers. During the transitions, the memory devices are inaccessible and processors are halted. We estimate this overhead mainly based on the runtime overhead of scaling DRAM frequency that is 512 memory cycles and 28 ns [7]. The overhead is estimated to be 0.67 us given the 800MHz memory frequency.

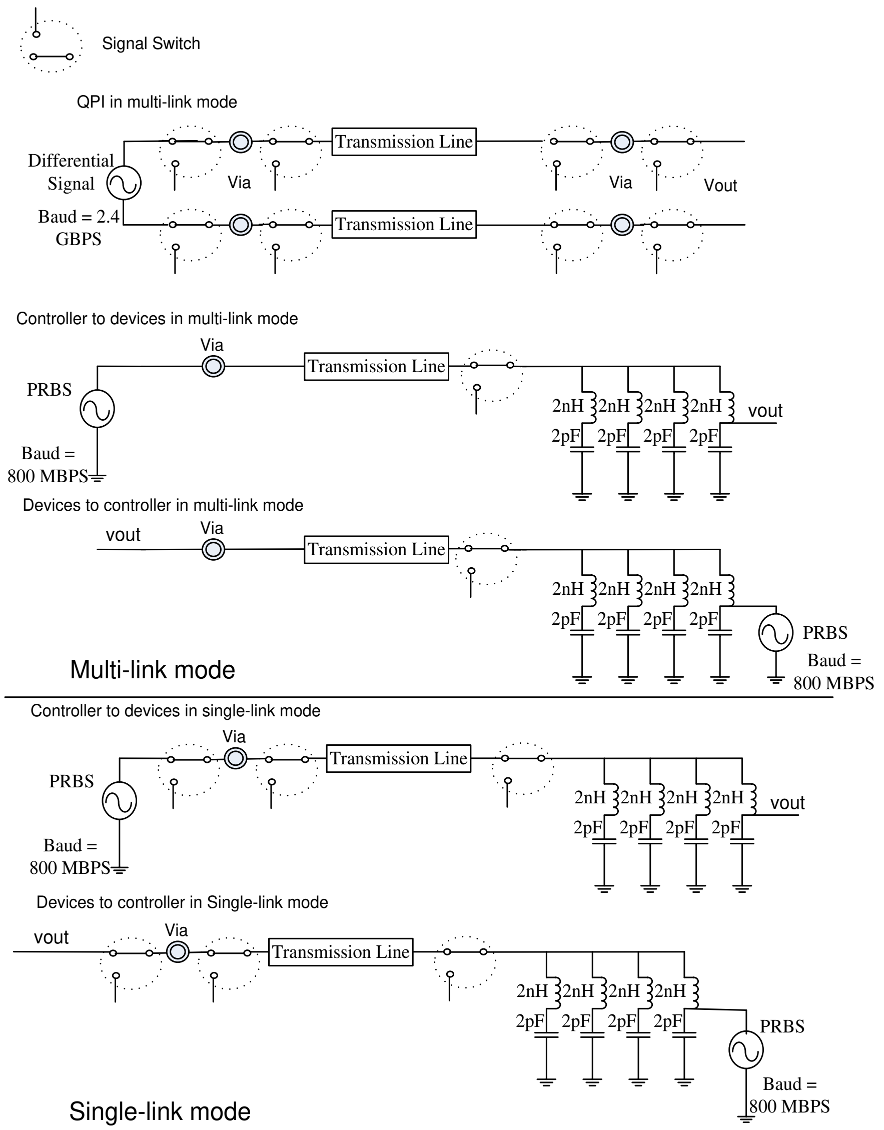
Additionally, we commit all the memory requests in the queue before turning off/on a memory channel. Given the read and write request queues in a memory channel have 32 total entries and each request takes 40ns, the runtime overhead of turning on/off a memory channel can be estimated to be 1.28us which is still affordable. The total overhead of turning off/on memory buses is 1.95us.

The latter comes from re-stabilizing the signals on the QPI buses and turning on/off the QPI PHYs. Note that the processors are not halted but cannot use the extra bandwidth of from QPI buses during the transitions when the systems are switching to multi-link mode. We conservatively estimate that switching QPI buses takes the same amount of time as switching memory buses. So the total runtime overhead is estimated to be 3.13 us.

5.2.8 Signal integrity

We setup up Spice models in the two modes shown in Figure 5-5, and test the signal integrity with mentor graphic tools [11] to prove that our design maintains signal integrity for the data path signals on memory buses and the data lane signals on the QPI bus. Memory bus signals are bi-directional with an 800 MHz frequency while the data lane on the QPI bus runs at a 2.4GHz frequency.

For multi-link mode, we show the eye diagram for the signal of a data lane on a QPI bus in Figure 5-6 (a). The eye diagram shows an open eye though it has some noise due to signal reflections. We also show the eye diagrams for a signal on the memory bus in Figure 5-6 (b) and (c). They also have open eyes though the signal in Figure 5-6 (b) suffers from signal reflections and the signal in Figure 5-6 (c) suffers from the loads of large capacity from memory devices. Additionally, we show the eye diagrams for single-link mode in Figure 5-6 (d) and (e) when data is read from memory devices or written into memory devices each having clearer eyes compared to Figure 5-6 (b) and (c) due to



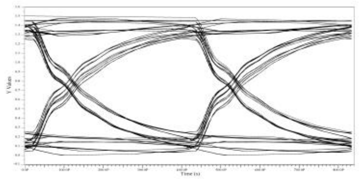
Signal Switch
QPI in multi-link mode
Differential Signal
Baud = 2.4 GBPS
Via
Transmission Line
Vout
Controller to devices in multi-link mode
PRBS
Baud = 800 MBPS
2nH
2pF
vout
Devices to controller in multi-link mode
Multi-link mode
Controller to devices in single-link mode
Devices to controller in Single-link mode
Single-link mode

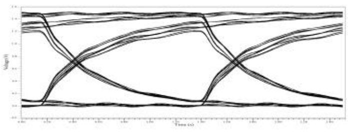
Figure 5-5. The Spice models for QPI buses and memory buses in single-link mode and the multi-link mode

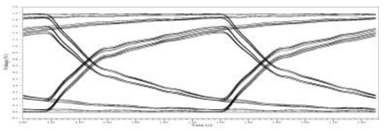
less signal switches on the paths. These figures indicate that acceptable signal quality is retained in both scenarios.

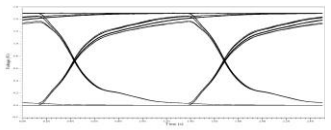
5.3. EXPERIMENTAL SETUP

We setup up our stimulated system with sniper 6.1 [26] using the system configuration shown in Table 5-1. The system has two processors with the configuration


(a). The eye diagram of a QPI bus in the multi-link mode


(b). The eye diagram of memory bus when reading data from devices in the multi-link mode


(c). The eye diagram of memory bus when writing data to devices in the multi-link mode


(d). The eye diagram of memory bus when reading data from devices in the single-link mode

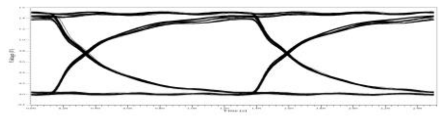

(e). The eye diagram of memory bus when writing data to devices in the single-link mode

Figure 5-6. Eye diagrams

based on the Intel Xeon X5550. Each processor has 4 memory channels and 1 QPI bus while in single-link mode, and then has 2 memory channels and 3 QPI buses while in multi-link mode. The energy consumption is estimated via the McPAT tool [56]. We also list the selected multi-thread workloads shown in Table 5-2 as well as the number of un-core requests per instruction, and the percentage of QPI latencies per the total un-core latencies. The workloads are selected from NPB benchmark [69], Splash2 benchmark

[87], and Decapo benchmark [25]. Since the lusearch workload involves a considerable number of system calls, we run it in jikes RVM (Research Virtual Machine) [18] on the sniper simulator. We separate the workloads into those workloads exhibiting intensive inter-socket communication and those workloads showing moderate or low inter-socket communication.

Most experiments are conducted using the communication intensive workloads which reveal the benefits of multi-link mode, while we use the non-intensive workloads to compare the performances of static and dynamic switching. We fast forward workloads into selected regions that show intensive inter-socket traffic, and then warm up the cache for 1 billion instructions. We run total 800 million instructions for each

Table 5-1. The configuration of the simulated system

Component	Parameters
system	two processors
Processor	4 cores
Core	2.66 GHz, 4-way issue, 128-entry ROB hybrid local/global predictor
Cache Line Size	64B, LRU replacement
L1-I	32KB, 4 way, 4 cycle access time
L1-D	32KB, 8 way, 4 cycle access time
L2 cache	256 KB per core, 8 way, 8 cycle
L3 cache	shared 8 MB, 16 way, 30 cycle
Coherence protocol	MSI
DRAM	line-interleaved mapping, 34.1GB/s
DRAM cache	128 MB, 16 way, 512GB/s
QPI bus	20 link width, 3.2GHz, 25.6 GB

workload since some threads, e.g. the garbage collector, run much fewer instructions than others in the workload lusearch. We also run each workload five times and show the 95% confidence intervals for performance comparisons.

5.4. RESULT

5.4.1 Performance of the static switching

We evaluate the performances of the baseline, in which the system runs in single-link mode; static switching; and dynamic switching. Figure 5-7 shows the results of static switching and dynamic switching, which are normalized against the results of the

Table 5-2. The selected workloads

workloads	benchmark suit	Un-core request per 1K inst.	The percentage of QPI latencies
Workloads with intensive inter-socket traffic			
bt	NPB	3.70	77%
cg	NPB	20.44	62%
is	NPB	20.54	30%
lu	NPB	3.51	72%
sp	NPB	10.52	86%
ua	NPB	4.15	74%
ocean	Splash2	13.87	99%
lusearch	Decapo	8.92	95%
Workloads with moderate and low inter-socket traffic			
ft	NPB	6.89	61%
mg	NPB	15.36	30%
fmm	Splash2	0.15	27%
radiosity	Splash2	0.53	35%
raytrace	Splash2	0.95	25%

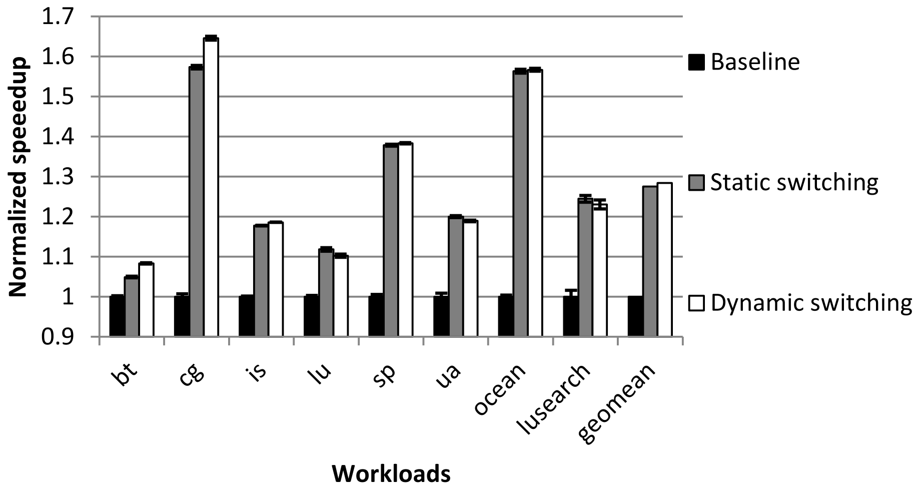


Figure 5-7. The normalized speedup of the static switching and the dynamic switching compared with the baseline

baseline for each workload. Static switching and dynamic switching gain a performance improvement of 28% and 29% respectively compared to the baseline. We also show the reduced latencies of un-core requests in static switching normalized against that in the baseline shown in Figure 5-8. Note the latencies only account for the latencies incurred outside the cores. The workloads cg and ocean achieve speedups of 1.54 and 1.55 respectively, which are much more than other workloads, since they have intensive un-core traffic and their un-core latencies are significantly reduced by multi-link mode. The other workloads gain moderate performance improvements. For example, is also has intensive un-core traffic but sees a smaller reduction of the latencies, while *lusearch* has a significant reduction of the latencies but moderate un-core traffic.

5.4.2 Performance of the dynamical switching

Dynamic switching can gain a similar performance improvement for the workloads in Figure 5-7 since it can detect phases of intensive inter-socket

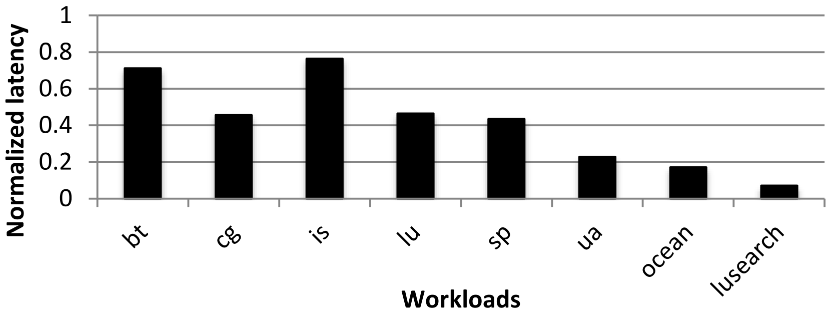


Figure 5-8. The latency of un-core requests for the static switching normalized against that of the baseline

communication. For example, dynamic switching improves the performance of the cg workload from 1.57 to 1.64, since it finds a period of low inter-socket traffic and guides the system switches back to the multi-link mode. Dynamic switching provides approximately the same performance improvements as static switching for workloads that have only a few intervals of low inter- socket traffic. We also list the number of the intervals that the system is in multi-link mode or in single-link mode in Table 5-3, as well as the number of times that the system switches to multi-link mode or single-link mode. The extra benefits of dynamic switching for the cg workload come from the system being guided back to single-link mode when it catches a consecutive series of 12 intervals in which the system exhibits low inter-socket communication but moderate local traffic shown in Figure 5-8.

We also test the performance of dynamic switching compared with static switching for workloads exhibiting moderate or low inter-socket traffic shown in Figure 5-9. The results are normalized against the performance of the baseline for each workload respectively. The dynamic switching gains are a normalized speed up of 1.18, 1.04 respectively for the ft, mg workloads, while static switching only achieves a normalized

Table 5-3. The intervals in the multi-link mode and in the single-link mode as well as the times of switching to the multi-link mode and the single-link mode

	The multi-link mode	The single-link mode	The switching to the multi-link mode	The switching to the single-link mode
Workloads of intensive inter-socket traffic				
bt	235	32	2	1
cg	539	12	2	1
is	451	1	1	0
lu	335	7	2	1
sp	403	2	2	1
ua	354	9	4	3
ocean	465	2	2	1
lusearch	463	4	4	3
Workloads of moderate and low inter-socket traffic				
ft	209	83	3	2
mg	570	185	7	6
fmm	135	1	1	0
radiosity	524	27	18	17
raytrace	1577	19	18	17

speedup of 0.84, 0.87. Dynamic switching captures several stable periods in which performance can be improved via switching the system back to single-link mode. Dynamic switching suffers from spikes of intensive local traffic that are hardly captured and thus only achieves a speedup of 0.91 for radiosity, which is close to the performance of static switching. Even though, the proposed dynamic qSwitch can still achieve a geometric mean of 1.02 for the five benchmarks with moderate or low inter-socket traffic.

5.4.3 Energy efficiency

We investigate the energy consumption of static switching which is normalized against that of the baseline for each workload respectively shown in Figure 5-10. Static switching reduces the average energy consumption by 12% in geometric mean since it

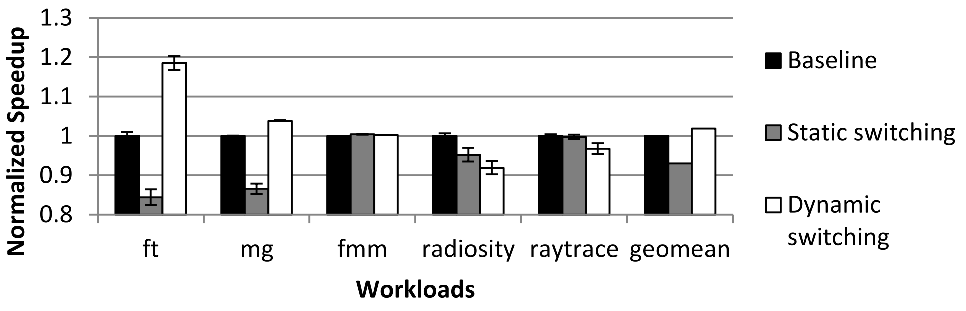


Figure 5-9. The normalized speedup of the static switching and the dynamic switching compared with baseline for the workloads with moderate or low inter-socket traffic

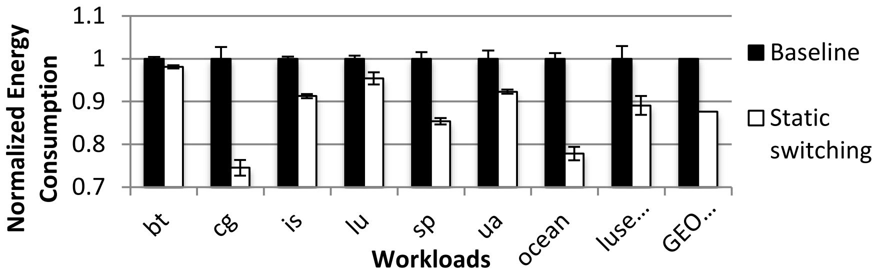


Figure 5-10. The energy consumption in the static switching normalized against the baseline

improves the system performance with minimal energy overhead. The more performance improvement that multi-link mode has gained, the more energy consumption is reduced. For example, the workloads cg and ocean save 25% and 22% more energy than the others, while they also achieve more performance benefits compared to others.

5.4.4 Enhancement from a stride prefetcher

We investigate the performances of static switching and the baseline combined with a stride prefetcher shown in Figure 5-11. The results are normalized against the baseline without a prefetcher and we show the performances with prefetchers that have a prefetch degree of 1, 2, and 4, which denotes the number of prefetches issued on every memory reference. The performances of the baseline with the prefetchers are 1.0, 1.1, 1.11 respectively in the geometric mean, while the performances of static switching with the prefetcher are 1.28, 1.53, and 1.55 respectively. Static switching with a prefetching degree of 2 shows a considerable performance improvement compared to that with a prefetching degree of 1. The aggressive prefetchers shift the performance bottlenecks toward the QPI especially for the is workload, which can be verified in Figure 5-12 which reveals the percentage of the total un-core latencies made up by QPI latencies for the baseline with prefetchers. This percentage of latency for the is workload is increased significantly when the prefetching degree is increased from 1 to 2, since the prefetcher now exploits the high bandwidth of the DRAM cache via increasing the memory level parallelism and thus reduces DRAM cache latencies but suffers from a limited QPI bandwidth which offset the benefit of memory level parallelism. Prefetchers boost the percentage of QPI latencies for the workload since its un-core latencies in the DRAM cache are considerable even running on the baseline without a prefetcher compared with other workloads.

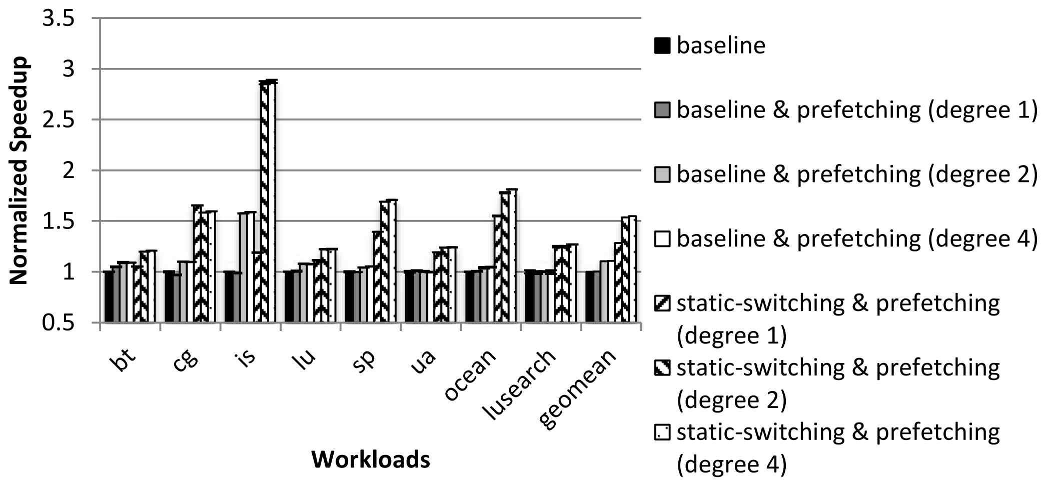


Figure 5-11. The normalized speedup of the static switching with a prefetcher (degree 1, 2, 4) compared with baseline and the prefetcher

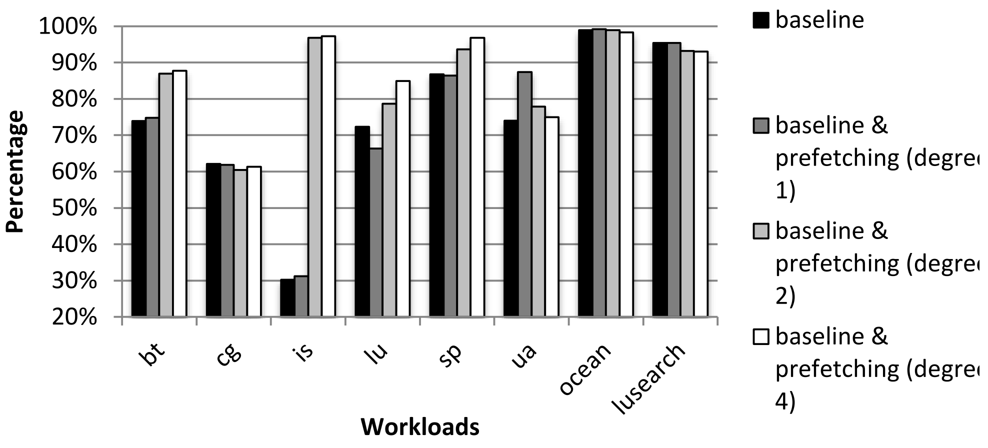


Figure 5-12. The ratio between the un-core latencies of QPI and the total un-core latencies with the baseline and a prefetcher (degree 1, 2, 4)

We also evaluate the performance of static switching using different configurations of the DRAM cache. This work heavily relies on the DRAM cache's superior bandwidth compared to off-chip main memory devices and the QPI and thus we want to verify the substantial benefit of the static switching in the broad design space of the DRAM cache.

5.4.5 The bandwidth of the DRAM cache

We investigate the performance improvement of static switching with different bandwidths of DRAM caches shown in Figure 5-13. We vary the bandwidths from 128GB/s to 1024GB/s and compare the performances of static switching and the baseline with the same DRAM cache bandwidth. Figure 5-13 shows the performance of static switching normalized against the performance of the baseline accordingly. The performance improvements are 1.27, 1.28 ,1.28 , and 1.29 in the geometric mean for the DRAM cache bandwidths of 128GB/s, 256GB/s, 512GB/s, and 1024GB/s respectively. The relatively stable improvements indicate that the benefit of static switching is

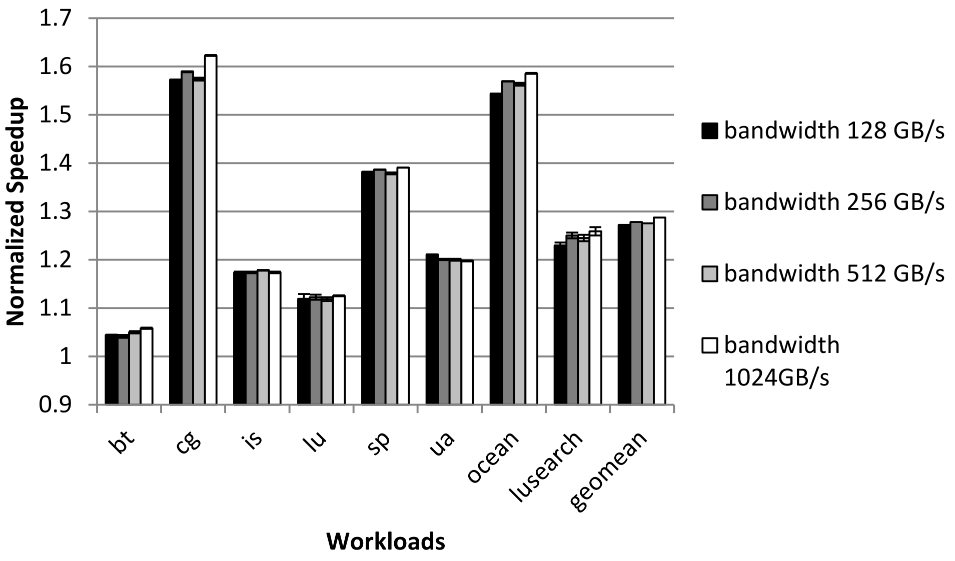


Figure 5-13. The normalized speedup of the static switching with the different bandwidths of DRAM cache

consistent as long as the bandwidth of cache DRAM is larger than that of the main memory.

5.4.6 The size of DRAM cache

We also evaluate the performance improvement of static switching with different sizes of DRAM cache (32MBytes, 64MBytes, and 128MBytes). When the size is increased, more un-core requests hit the DRAM cache that has much more bandwidth compared to off-chip memory devices. On the other hand, the smaller DRAM cache size will decrease its performance impact on the performance. Static switching achieves an average of 20%, 21% and 28% performance improvements in geometric mean, which are normalized against the performance in the baseline with the same DRAM cache size respectively shown in Figure 5-14. The performance improvements for most workloads increase slightly as the size of DRAM cache is increased, while the performance

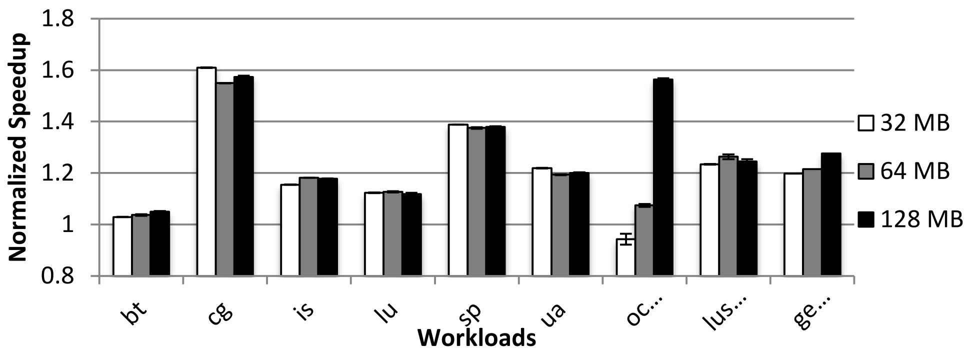


Figure 5-14. The normalized speedup of the static switching with the different sizes of DRAM cache

improvement of the workload ocean increases quickly from -6% to 56% due to more remote requests hitting DRAM caches, which can be verified by the reduced latencies of un-core requests for ocean in Figure 5-15. It also shows the latencies of un-core requests in the multi-link mode normalized against the latencies in the single-link mode. The figure shows most workloads slightly reduce the latency of un-core requests as the DRAM cache size is increased, while the latencies for ocean are reduced from 1.58 to 0.136. Decreasing the size of the DRAM cache from 128MBytes to 32 MBytes causes the percentage of the latencies of the QPI in the total un-core latencies to drop from 98% to

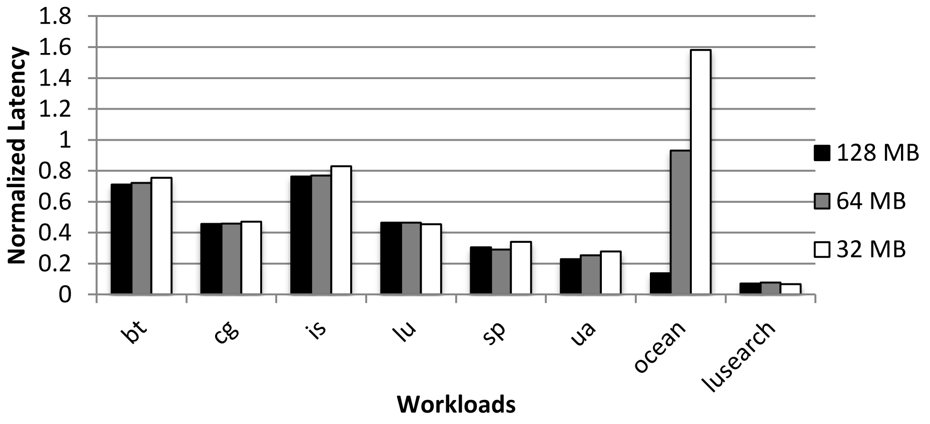


Figure 5-15. The normalized latencies of un-core requests in the static switching with the different sizes of DRAM cache

14% for the baseline running the ocean workload, while the percentage of the latencies of the off-chip main memory in the total un-core latencies increase from 0.002% to 86% for the baseline running with the same workload.

5.4.7 The frequency of QPI buses

We investigate the performance improvement of static switching with different QPI bus frequencies (2.4GHz, 3.2GHz, and 4.8GHz). 2.4GHz is the lowest frequency of the QPI buses, while 4.8GHz is first introduced on Hashwell-E/EP platform. Boosting the frequency of QPI buses can gain more bandwidth but incurs higher power consumption and poses more difficulties for routing QPI traces on the motherboard. Figure 5-16 shows the performance improvement of static switching normalized against the performance of the baseline with different QPI bus frequencies. Static switching with QPI frequencies 2.4GHz, 3.2GHz and 4.8GHz achieve the average speedups of 1.44, 1.28, and 1.15 in geometric mean respectively. Our design can still gain a moderate performance improvement with the high frequency of 4.8GHz and can significantly increase the performance improvement with the low 2.4GHz frequency. Figure 5-17 shows the ratio between the un-core latencies in QPI and the total un-core latencies with the baseline

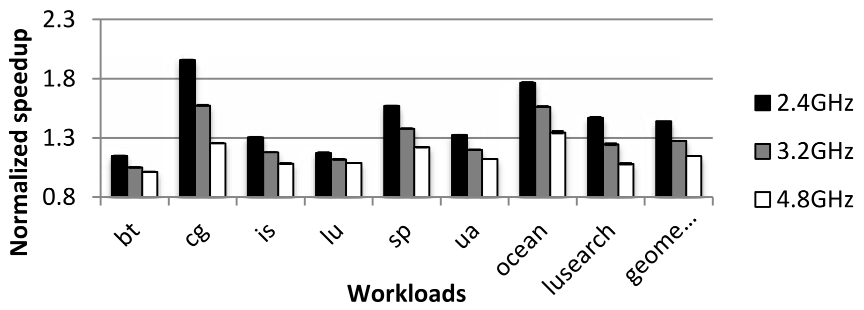


Figure 5-16. The normalized speedup of the static switching with the different frequencies of QPI

with the different QPI bus frequencies. The ratio of QPI decreases as the frequency of QPI buses increases, which shortens the time of transferring data over the QPI buses and the waiting time of packets in the QPI. The ratio for the cg workload drops from 79% to 43% as the frequency of the QPI buses increase from 2.4GHz to 4.8GHz, while the performance benefit of the static switching decreases from 1.96 to 1.25. The 4.8GHz frequency of the QPI buses increases the percentage of the latencies from the DRAM cache with respect to the total un-core latency. For example, the frequency increases the percentage from 17% to 48% for the cg workload when the frequency of the QPI buses increases from 2.4GHz to 4.8GHz.

5.5. CONCLUSION

Multi-socket systems are widely used for massive parallel workloads to improve throughput. The performance of multi-socket system suffers from limited off-chip bandwidth confined by the scarce resource of processor pins. This problem can be

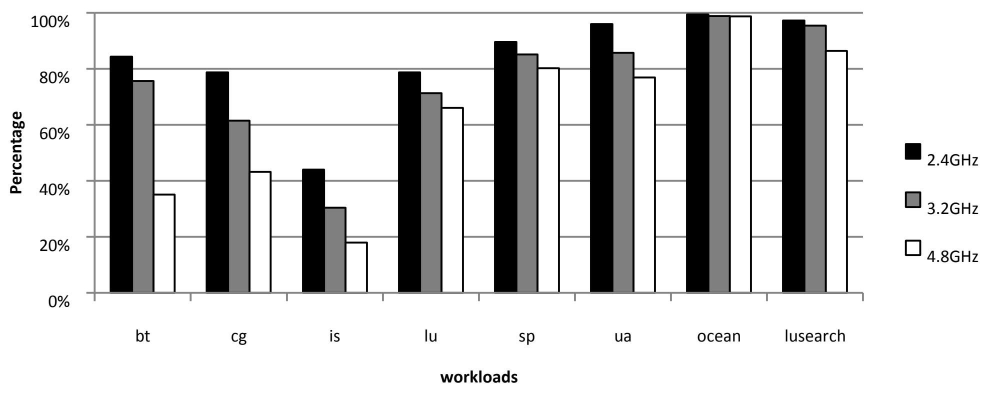


Figure 5-17. The ratio between the un-core latencies of QPI and the total un-core latencies with the baseline and the different frequencies of QPI buses

relieved by the DRAM cache that is introduced to reduce the long latency of off-chip access via providing a large space to hold data and superior bandwidth to reduce queuing delay. The DRAM cache reduces the latencies of accessing main memory as the main contributor of the un-core latencies, while the latencies of inter-socket communication emerge as a considerable bottleneck for workloads that frequently fetch data from remote memory.

qSwitch is proposed to reduce the inter-socket latencies at the cost of local memory bandwidth, since the DRAM cache significantly reduces the number of off-chip local requests and thereby the local memory bandwidth becomes over-sufficient in some cases. We design qSwitch from the off-chip bus connection to the switching agents in order to smoothly switch the system between the two modes. We investigate the signal integrity and discuss the design overhead to verify its feasibility. We also evaluation the performance benefits of qSwitch using different configurations of the DRAM cache and QPI to show the benefits exist in a broad design space.

This work identifies the latency of inter-socket communication as one of the performance bottlenecks in the era of DRAM cache for massive parallel workloads. It implies that the performance of the workloads can be improved via the optimization of inter-socket communication such as wisely scheduling remote requests or reducing unnecessary remote requests. Furthermore, the limited bandwidth of inter-socket communication could become increasingly painful as the number of cores on a die increases and more cores share bandwidth. Scaling the inter-socket bandwidth with the number of cores is likely to be a challenge in the near future.

5.6. SUMMARY

In this study, we majorly descript the two works based on the switchable pins for the off-chip bandwidth and mitigating dark silicon. The limited off-chip memory bandwidth has been widely acknowledged as a major constraint to prevent us from obtaining commensurate performance benefit from the faster processor cores. This is especially challenging in the current multi-core era due to a high volume of memory requests coming from an increasing number of processor cores. To alleviate the shortage of off-chip bandwidth, we propose an innovative pin switching technique which dynamically allocates pins for power delivery or signal transmission with minimal changes to the circuit. By accurately identifying memory-intensive phases at runtime, the proposed strategy converts a portion of the pins used for power delivery to signal transmission mode, providing additional off-chip bandwidth and improving the overall performance. As shown by the evaluation results, along with other techniques including Dynamic Switching and stride prefetching, our scheme is capable of significantly accelerating the program execution.

Dark silicon is gradually becoming a daunting conundrum that threatens the scaling of Moore's Law in the future, with the stall of Dennard scaling. While thermal constraint are widely believed to be the main cause of this phenomenon, the limited number of pins on the chip package also confines the maximum number of simultaneously active transistors, thus preventing us from obtaining a sufficient performance improvement by increasing transistor density. To mitigate this limitation, we propose a novel mechanism to dynamically switch a portion of I/O pins to power pins in order to light up dark silicon by delivering extra power. We also employ an advanced statistical model to train a prediction model that can be employed by the OS to govern the pin switching. Our evaluation results demonstrate that the proposed pin switching mechanism can remarkably enhance the overall performance compared with conventional designs.

We also present two challenging and meaningful yet research topic based on the underlying idea of switchable pins: boosting off-chip bandwidth with PCM and increasing inter-socket bandwidth via switchable pins. The topics are expected to lead us to explore the benefits of the switchable pins on the two areas.

REFERENCES

[1] 4thgeneration core family desktop http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/4th-gen-core-family-desktop-vol-1-datasheet.pdf

[2] 4-bit parallel-to-serial converter http://www.micrel.com/_PDF/HBW/sy10-100e446.pdf

[3] 4-bit serial-to-parallel converter http://www.micrel.com/_PDF/HBW/sy10-100e445.pdf.

[4] http://opencores.org

[5] http://www.cadence.com/products/ld/rtl_compiler/pages/default.aspx

[6] https://www.si2.org/openeda.si2.org/projects/nangatelib

[7] http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/4th-gen-core-family-desktop-vol-1-datasheet.pdf

[8] http://www.freescale.com/files/32bit/doc/app_note/AN3940.pdf

[9] http://www.protoexpress.com/content/stcapability.jsp

[10] http://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool

[11] http://www.mentor.com/products/ic_nanometer_design/analog-mixed-signal-verification/eldo/.

[12] http://www.itrs.net/Links/2012ITRS/2012Tables/AssemblyPkg_2012Tables.xlsx.

[13] HotSpot. http://lava.cs.virginia.edu/HotSpot/

[14] COMSOL Multiphysics. http://www.comsol.com/

[15] International Technology Roadmap for Semiconductors, ITRS 2006 Update; see http://www.itrs.net/Links/2006Update/2006UpdateFinal.htm

[16] Intel Technology Roadmap for Semiconductors: Process Integration, Devices, and Structures, Semiconductor Industry Assoc. 2007; http://www.itrs.net/Links/2007ITRS/2007_Chapters/2007_PIDS.pdf.

[17] Intel Xeon Processor E5-2450L. http://ark.intel.com/products/64610/Intel-Xeon-Processor-E5-2450L-20M-Cache-1_80-GHz-8_00-GTs-Intel-QPI.

[18] Jikes RVM. http://www.jikesrvm.org

[19] Jung Ho Ahn, Norman P. Jouppi, Christos Kozyrakis, Jacob Leverich, and Robert S. Schreiber. 2009. Future scaling of processor-memory interfaces. In Proceedings of the

Conference on High Performance Computing Networking, Storage and Analysis (SC '09).

[20] An Introduction to the Intel® QuickPath Interconnect, Intel, 2009.

[21] H. Barowski, T. Brunschwiler, H. Harrer, A. Huber, B. Michel, T. Nigemeier, S. Paredes, and J. Supper. Heat sink integrated power delivery and distribution for integrated circuits. Patent Application Publication. Publication number: US 2012/0106074 A1.

[22] S. Beamer, C. Sun, Y. Kwon, A. Joshi, C. Batten,V. Stojanovic, K. Asanovic, Re-Architecting DRAM Memory Systems with Monolithically Integrated Silicon Photonics, In Proceeding of ISCA, 2010.

[23] R. Bitirgen, E. Ipek, and J. F. Martinez. Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach. In MICRO, 2008.

[24] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 simulator. SIGARCH Comput. Archit. News 39, 2 (August 2011), 1-7.

[25] S. M. Blackburn, R. Garner, C. Hoffman, A. M. Khan, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, , J. E. B. Moss, A. Phansalkar, D. Stefanovic, T. VanDrunen, , D. von Dincklage, B. Wiedermann, The DaCapo Benchmarks: Java Benchmarking Development and Analysis, In Proceeding of OOPSLA 2006.

[26] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Ex- ploring the level of abstraction for scalable and accurate parallel multi-core simulations. In Proceedings of SC, 2011.

[27] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron. 2009. Rodinia: A Benchmark Suite for Heterogeneous Computing. In Proceedings of the IEEE International Symposium on Workload Charac-terization (IISWC), pp. 44-54, Oct. 2009.

[28] S. Chen, Y. Hu, Y. Zhang, L. Peng, J. Ardonne, S. Irving, and A. Srivastava, "Increasing Off-Chip Bandwidth in Multi-Core Processors with Switchable Pins," In Proceedings of The ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), Minneapolis, MN, Jun. 2014.

[29] M. Chen, X. Wang, and X. Li. Coordinating Processor and Main Memory for Efficient Server Power Control. In ICS, 2011.

[30] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu. Memory power management via dynamic voltage/frequency scaling. In Proceedings of the 8th ACM international conference on Autonomic computing (ICAC '11).

[31] K. DeHaven, J. Dietz "Controlled collapse chip connection (C4)-an enabling technology," Electronic Components and Technology Conference, 1994. Proceedings, 44th, vol., no., pp.1,6, 1-4 May 1994.

[32] Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas F. Wenisch, and Ricardo Bianchini. 2012. CoScale: Coordinating CPU and Memory System DVFS in Server Systems. In MICRO '12.

[33] Y. Deng and J. Liu. Optimization and Evaluation of a High-Performance Liquid Metal CPU Cooling Product. In IEEE Transaction on Components, Packaging and Manufacturing Technology, 2013.

[34] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, R. Bianchini, MemScale: active low-power modes for main memory, in Proceeding of ASPLOS, 2011.

[35] L. Duan, B. Li, and L. Peng, Versatile prediction and fast estimation of architectural vulnerability factor from processor performance metrics, in HPCA 2009.

[36] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In ISCA 2011.

[37] J. Friedman and N. Fisher. Bump hunting in high-dimensional data. In Statistics and Computing, 9, 1999.

[38] N. Goulding-Hotta, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, P. Huang, M. Arora, S. Nath, V. Bhatt, J. Babb, S. Swanson, and M. B. Taylor. The greendroid mobile application processor: an architecture for silicon's dark future. IEEE Micro, 31(2):86-95, Mar-Apr 2011.

[39] G. Hamerly, E. Perelman, J. Lau, and B. Calder. SimPoint 3.0: Faster and More Flexible Program Analysis. In Workshop on Modeling, Benchmarking and Simulation, June 2005.

[40] N. Hardavellas, M. Ferdman, A. Ailamaki, and B. Falsafi. Power scaling: the ultimate obstacle to 1K-core chips. Technical report NWU-EECS-10-05, Northwestern University, 2010.

[41] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. IEEE Micro, 31(4):6-15, Jul-Aug 2011.

[42] Sorin Iacobovici, Lawrence Spracklen, Sudarshan Kadambi, Yuan Chou, and Santosh G. Abraham. 2004. Effective stream-based and execution-based data prefetching. In Proceedings of ICS '04.

[43] E. Ipek, O. Mutlu, J. Martinez, and R. Caruana. Self-Optimizing Memory Controllers: A Reinforcement Learning Approach. In Proceedings of ISCA, 2008.

[44] B. Jacob, S. W. Ng, and D. T. Wang. Memory Systems-Cache, DRAM, Disk. Elsevier, 2008.

[45] R. Jakushokas, M. Popovich, A.V. Mezhiba, S. Kose, and E.G. Friedman. Power Distribution Networks with On-Chip Decoupling Capacitors. 2011.

[46] D. Jevdjic, S. Volos, B. Falsafi, Die-Stacked DRAM Caches for Servers. In Proceedings of ISCA, 2013.

[47] Yoongu Kim, Vivek Seshadri, Donghyuk Lee, Jamie Liu, and Onur Mutlu. 2012. A case for exploiting subarray-level parallelism (SALP) in DRAM. SIGARCH Comput. Archit. News 40, 3 (June 2012).

[48] Y. Kim, D. Han, O. Mutlu, M. H.Balter, ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers. In Proceedings of HPCA, 2010.

[49] D. H. Kim, K. Athikulwongse, M. Healy, M. Hossain, M. Jung, I. Khorosh, G. Kumar, Y. Lee, D. Lewis, T. Lin, C. Liu, S. Panth, M. Pathak, M. Ren, G. Shen, T. Song, D. H. Woo, X. Zhao, J. Kim, H. Choi, G. Loh, H. Le, and S. K. Lim. 3D-MAPS: 3D Massively Parallel Processor with Stacked Memory. In ISSCC 2012.

[50] J. Kim, J. Shim, J. S. Pak, and J. Kim. Modeling of Chip-Package-PCB Hierarchical Power Distribution Network based on Segmentation Method. Advanced Packaging and Systems Symposium, 2008

[51] W. Kim, D. Brooks, and G. Wei. “A Fully-Integrated 3-Level DC/DC Converter for Nanosecond-Scale DVS with Fast Shunt Regulation,” IEEE International Solid-State Circuits Conference (ISSCC-11), Feb. 2011.

[52] K. L. Kishore and V.S.V. Prabhakar. VLSI Design, I K International Publishing House, 2009.

[53] A. V. Krishnamoorthy, R. Ho, X. Zheng, H. Schwetman, J. Lexau, P. Koka, G. Li, I. Shubin, J. E. Cunningham, Computer Systems Based on Silicon Photonic Interconnects, in Proceedings of the IEEE, vol.97, no.7, pp.1337-1361, July 2009.

[54] E. Kultursay, M. Kandemir, A. Sivasubramaniam, O. Mutlu. 2013. Evaluating STT-RAM as an energy-efficient main memory alternative. In Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on , vol., no., pp.256,267, 21-23 April 2013.

[55] BC, Lee; P, Zhou; J, Yang; Y, Zhang; B, Zhao; E, Ipek; O, Mutlu,; D, Burger, Phase-change technology and the future of main memory, IEEE Micro, vol 30 no. 1 (2010), pp. 131-141.

[56] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen and N. P. Jouppi. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In MICRO, Dec. 2009.

[57] Kai Ma, Xue Li, Ming Chen, and Xiaorui Wang. 2011. Scalable power control for many-core architectures running multi-threaded applications. In Proceedings of ISCA 2011.

[58] Krishna T. Malladi, Benjamin C. Lee, Frank A. Nothaft, Christos Kozyrakis, Karthika Periyathambi, and Mark Horowitz. 2012. Towards energy-proportional datacenter memory with mobile DRAM. In Proceedings of ISCA '12.

[59] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis. Power Management of Datacenter Workloads Using Per-Core Power Gating. In Computer Architecture Letter 2009.

[60] B. C. Lee, E. Ipek, D. Burger, and O. Mutlu, 2009. Architecting Phase Change Memory as a Scalable DRAM Alternative. In ISCA, 2009.

[61] Mentor Graphics SPICE Simulator ELDO.

[62] Micron Corp. Micron 2 Gb x 4, x8,x16, DDR3 SDRAM: MT41J512M4, MT41J256M4, and MT41J128M16, 2011.

[63] T. N. Miller, X. Pan, R. Thomas, N. Sedaghati, and R. Teodorescu. Booster: Reactive Core Acceleration for Mitigating the Effects of Process Variation and Application Imbalance in Low-Voltage Chips. In HPCA 2012.

[64] O, Mutlu. 2013. Memory scaling: A systems architecture perspective. In Memory Workshop (IMW), 2013 5th IEEE International, May 2013.

[65] O. Mutlu and T. Moscibroda. Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors. In Proceedings of MICRO, 2007.

[66] O. Mutlu and T. Moscibroda. Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems. In Proceedings of ISCA, 2008.

[67] Model for a 16nm, 0.9V process: http://ptm.asu.edu/modelcard/LP/16nm_LP.pm.

[68] Moinuddin K. Qureshi, Daniel N. Lynch, Onur Mutlu, and Yale N. Patt. 2006. A Case for MLP-Aware Cache Replacement. In ISCA '06.

[69] NAS Parallel Benchmarks, https://www.nas.nasa.gov/publications/npb.html.

[70] M. K. Qureshi, G. H. Loh. Fundamental Latency Trade-offs in Architecting DRAM Caches. In Proceeding of MICRO, 2012.

[71] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. K. Martin. Computational Sprinting. In HPCA 2012.

[72] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, K. Strauss, S. Sarangi, P. Sack, and P. Montesinos. SESC Simulator, January 2005.

[73] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin. Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling. In ISCA 2009.

[74] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. DRAMSim2: A Cycle Accurate Memory System Simulator. In IEEE Computer Architecture Letters 2010.

[75] Semiconductor Industry Assoc. 2007. "Int'l Technology Roadmap for Semiconductors: Process Integration, Devices, and Structures." Semiconductor Industry Assoc., Web. Dec. 2007.

[76] Standard Performance Evaluation Corporation. SPEC CPU 2006.

[77] Silicon Photonics: Challenges and Future, An OIDA Forum Report, OIDA, 2007.

[78] Standard Performance Evaluation Corporation 2001. " SPEC OMP 2001." Standard Performance Evaluation Corporation. Web. Jan. 2013.

[79] A. Snavely and D. M. Tullsen. Symbiotic job scheduling for a simultaneous multithreading processor. In ASPLOS-9, 2000.

[80] Y. H. Son, O. Seongil, Y. Ro, Jae W. Lee, J. H. Ahn. Reducing memory access latency with asymmetric DRAM bank organizations. In Proceedings of ISCA 2013.

[81] M. B. Taylor, Is dark silicon useful? In DAC, Jun. 2012.

[82] The 50G Silicon Photonics Link, intel Labs, 2010.

[83] Jose Tierno, Alexander Rylyakov, Daniel Friedman, Ann Chen, Anthony Ciesla, Timothy Diemoz, George English, David Hui, Keith Jenkins, Paul Muench, Gaurav Rao, George Smith III, Michael Sperling, Kevin Stawiasz. A DPLL-based per core variable frequency clock generator for an eight-core POWER7 x2122 microprocessor. In VLSIC. 2010.

[84] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous Multithreading: Maximizing On-Chip Parallelism. In ISCA 1995.

[85] Young Hoon Son, O. Seongil, Yuhwan Ro, Jae W. Lee, and Jung Ho Ahn. 2013. Reducing memory access latency with asymmetric DRAM bank organizations. In Proceedings of ISCA '13.

[86] Aniruddha N. Udipi, Naveen Muralimanohar, Niladrish Chatterjee, Rajeev Balasubramonian, Al Davis, and Norman P. Jouppi. 2010. Rethinking DRAM design and organization for energy-constrained multi-cores. In Proceedings of ISCA '10.

[87] S. C. Woo, M. Ohar, E. Torrie, J. P. Singh, A. Gupta, The SPLASH-2 Programs: Characterization and Methodological Considerations. In Proceeding of ISCA, 1995.

[88] Doe Hyun Yoon, Jichuan Chang, Naveen Muralimanohar, and Parthasarathy Ranganathan. 2012. BOOM: enabling mobile memory based low-power server DIMMs. SIGARCH Comput. Archit. News 40, 3 (June 2012).

[89] W. Zhang and T. Li. 2009. Exploring Phase Change Memory and 3D Die-Stacking for Power/Thermal Friendly, Fast and Durable Memory Architectures. In Proceedings of the 2009 18th International Conference on Parallel Architectures and Compilation Techniques (PACT '09).

[90] R. Zhang, B.H.Meyer, W. Huang, K. Skadron, and M.R. Stan. Some limits of power delivery in the multicore era. In Proceedings of the Workshop in Energy Efficient Design (WEED), in conjunction with ISCA 2012.

[91] R. Zhang, K. Wang, B. H. Meyer, M. Stan, and K. Skadron, "Architecture Implications of Pads as a Scarce Resource," In Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA), Jun. 2014.

[92] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu. 2008. Mini-rank: Adaptive DRAM architecture for improving memory power efficiency. In Proceedings of MICRO 2008.

[93] Hongzhong Zheng, Jiang Lin, Zhao Zhang, and Zhichun Zhu. 2009. Decoupled DIMM: building high-bandwidth memory system using low-speed DRAM devices. In Proceedings of ISCA '09.

[94] Hongzhong Zheng, Jiang Lin, Zhao Zhang, Eugene Gorbatov, Howard David, and Zhichun Zhu. 2008. Mini-rank: Adaptive DRAM architecture for improving memory power efficiency. In MICRO 41.

VITA

Shaoming Chen was born in 1985, in Wuhan, Hubei, China. He received his Bachelor of Engineering and Master of Engineering degrees in Electronics and Information Engineering from Huazhong University of Science and Technology, Wuhan, China, respectively in June 2008 and May 2012. Since then, he has been enrolled in the Department of Electrical and Computer Engineering at Louisiana State University, Baton Rouge, Louisiana, to pursue his doctorate degree. During this period, he passed his qualify exam in Fall 2012 and general exam in March 2015, respectively.