Using Switchable Pins to Increase Off-Chip Bandwidth in Chip-Multiprocessors

Shaoming Chen, Samuel Irving, Lu Peng, Yue Hu, Ying Zhang, and Ashok Srivastava

Abstract—Off-chip memory bandwidth has been considered as one of the major limiting factors of 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 observe that a processor requires much less power during memory intensive stages than is available. This is due to the fact that the frequencies of processor cores waiting for data to be fetched from off-chip memories can be scaled down in order to save power without degrading performance. Motivated by this observation, we propose a dynamic pin switching technique to alleviate this bandwidth limitation. This technique is introduced to dynamically exploit surplus power delivery pins to provide extra bandwidth during memory intensive program phases, thereby significantly boosting performance. This work is extended to compare two approaches for increasing off chip bandwidths using switchable pins. Additionally, it shows significant performance improvements for memory intensive workloads on a memory subsystem using Phase Change Memory.

Index Terms—Multiprocessors; Reconfigurable hardware

1 INTRODUCTION

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's performance is highly likely to be bottlenecked by slow memory accesses. An intuitive solution to this problem is to increase off-chip memory bandwidth by enabling more memory channels. Figure 1 illustrates the variation of normalized throughput with the number of memory channels increased from 1 to 4 when four 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.

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 an accurate memory-intensive phase identifier. Pins can be dynamically purposed for power delivery or signal transmission via accessory circuits. These circuits enable pins to deliver either quality power or signal and require relatively low area overhead. The total number of pins is growing slowly and becoming a performance bottleneck for both off-chip I/O and the power delivery network [42]. Switchable pins can be used to increase off-chip bandwidth without extra package costs, and are more cost-effective than simply increasing the total number of pins.

We identify memory-intensive phases by measuring key performance metrics at runtime. Extra off-chip bandwidth is demanded during phases with high memory intensity. Therefore, by configuring switchable pins to provide additional bandwidth for off-chip memory transactions, the performance of memory-intensive stages can be boosted. The novelty of our design lies in that we improve the performance of memory intensive workloads with enormous off-chip traffic without adding extra processor pins or additional package costs. This design is orthogonal to most existing works listed in Section 2 that improve the performance of memory intensive workloads and thus can yield performance improvement when used in conjunction with them.

Furthermore, we also investigate the performance benefits of our design [13] with a memory subsystem using Phase Change Memory (PCM).

In general, the main contributions of this paper are as follows:

1. We devise a memory controller that can dynamically increase the off-chip bandwidth at the cost of a lower core frequency. Results show a significant increase in throughput for memory-intensive workloads with only a slight hardware overhead.

- *This work is supported in part by NSF grants CCF-1017961 and CCF-1422408.*
- *S. Chen, S. Irving, L. Peng, Y. Hu, and A. Srivastava, Division of Electrical & Computer Engineering, School of Electrical Engineering and Computer Science, Louisiana State University, E-mail: {schen26, sirvin1,lpeng, yhu14, eesriv}@lsu.edu;*
- *Y. Zhang, Intel Corporation in Santa Clara, CA. E-mail: yzhan29@lsu.edu*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2546246, IEEE Transactions on Parallel and Distributed Systems

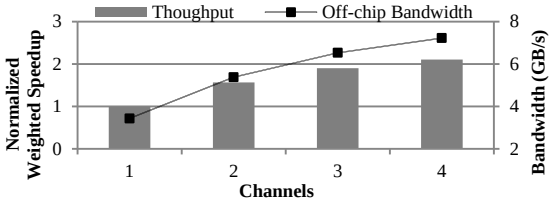


Figure 1. Normalized weighted speedup and off-chip bandwidth of 4 *lbm* benchmarks co-running on a processor with 1,2,3,4 memory channels

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

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

2. We propose a switchable pin design which can convert a power pin to a signal pin or the other way around. Detailed examinations at both the circuit and architectural level are conducted to validate the feasibility of the proposed design.
3. We examine the performance improvement of our design in various memory configurations. A sensitivity study is conducted to compare the benefits of our design using a different number of channels, buses, banks and ranks.
4. We investigate two approaches for increasing off-chip bandwidth using switchable pins denoted as "multi-bus mode" and "wide-bus mode". We detail the off-chip bus connections of the two modes and compare their performances.
5. We design Dynamic Switching to alleviate the negative side-effects of pin switching by actively identifying memory-intensive phases and only switching when certain conditions are 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.
6. 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.

2 Related Work

DRAM-Based Memory System: Several papers propose to physically alter the main memory in a DRAM-based memory system to improve performance and energy efficiency. Zheng 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 [44]. Papers also explore using low power DDR2 (LPDDR2) memory, in place of conventional DDR3, due to its higher energy efficiency [24][41].

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 [45]. Rank sub-setting is also proposed to improve the reliability and performance of a memory system [9].

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 [20]. An asymmetric DRAM bank organization can be used to reduce the bank access latency and improve the system performance [39]. Unlike preceding work, we focus on increasing off-chip bandwidth to boost the performance of the memory system as it is the major bottleneck in memory systems of the multi-core era.

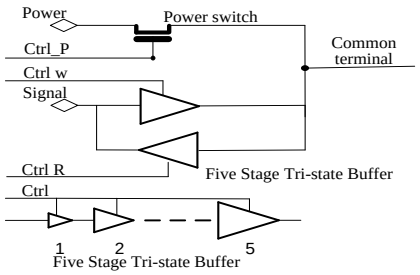


Figure 2 (a). The circuit of a signal-to-power switch

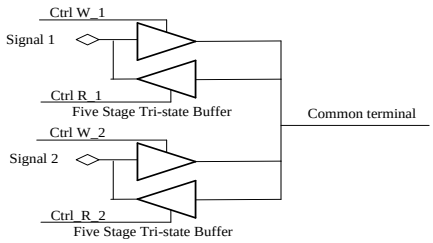


Figure 2(b). The circuit of a signal switch

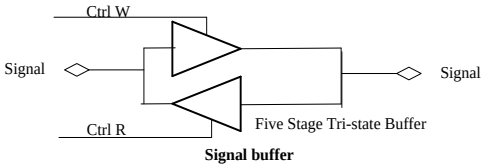


Figure 2(c). The circuit of a signal buffer

Off-Chip Bandwidth: Rogers et al. have already stressed the significance of off-chip bandwidth [33]. 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 [40]. Ipek designs a memory scheduler using principles of reinforcement learning to understand program behaviors and boost performance [10]. Mutlu et al. focus on boosting multi-threaded performance by providing fair DRAM access for each thread in their memory scheduler [30][31]. 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 [10][14][16]. However, few of them are able to increase the off-chip bandwidth beyond the constraint of static pin allocation.

Compared to previous works, our work proposes to dynamically increase off-chip bandwidth to boost the performance of memory intensive workloads. This work is orthogonal to others' and can be used in conjunction with them to provide additional performance gains. The main challenges of this work are modifying the system architecture and identifying memory-intensive phases. The architecture modifications involve additional circuits attached to switchable pines and modifying memory controllers to use the extra memory bandwidth. This design requires an effort to integrate multiple pieces

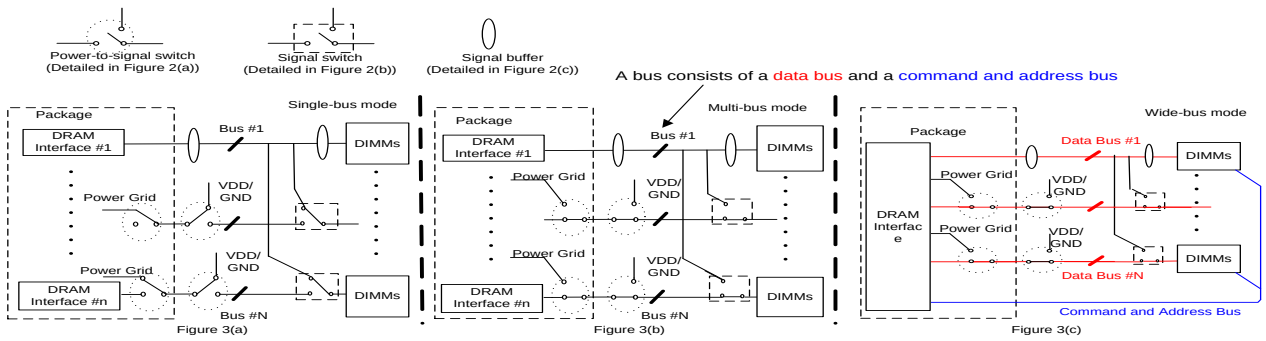


Figure 3. Overview of the hardware design of the off-chip bus connection for switching between single-bus mode and multi-bus mode or wide-bus mode

together, and an algorithm to exploit the benefits and offset the drawbacks.

3 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.

3.1 Pin Switch

Figure 2 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(a) is key to alternate a regular pin between the two modes. As illustrated in this figure, we utilize a dedicated power switch [27] which sits on the power delivery path to minimize the corresponding IR drop and power consumption with its 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 [27]. 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 [5]. 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(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(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. As the area of a signal switch is less than 62.3 um^2 based on 45nm technology with Cadence tools, the area overhead of the switches is negligible.

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 [5] shown in Table 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. After a sensitivity study in Section 4.1, in our experiment we will allocate at most three additional buses via pin switching because adding more leads to a considerable drop in performance.

3.2 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 single-bus mode shown in Figure 3(a) and the multi-bus mode shown in Figure 3(b) or the wide-bus mode shown in Figure 3(c). Single-bus mode is the default mode in which the system stays when turned on, while multi-bus and wide-bus modes are used to increase the off-chip bandwidth and boost the performance of memory intensive workloads using switchable pins.

Multi-bus mode uses switchable pins to add more memory buses which consist of data buses and buses of command and address. The single-bus mode can only access DRAM by a single bus, while the multi-bus mode can access them via the individual buses. 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.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2546246, IEEE Transactions on Parallel and Distributed Systems

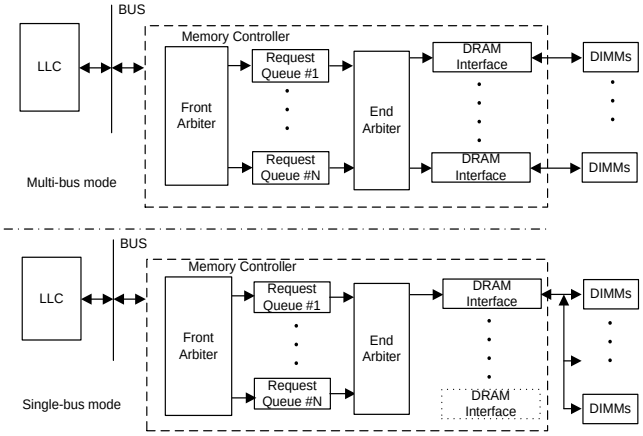


Figure 4. Overview of the hardware design of memory controller for switching between Multi-bus mode and Single-bus mode

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.

In order to implement the mechanism, we control the sig nal-to-power switch detailed in Figure 2(a) and the signal switch detailed in Figure 2(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 3 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 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 [7] and the DDR3 PCB layout guidelines [6], 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.

3.3 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 4. 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

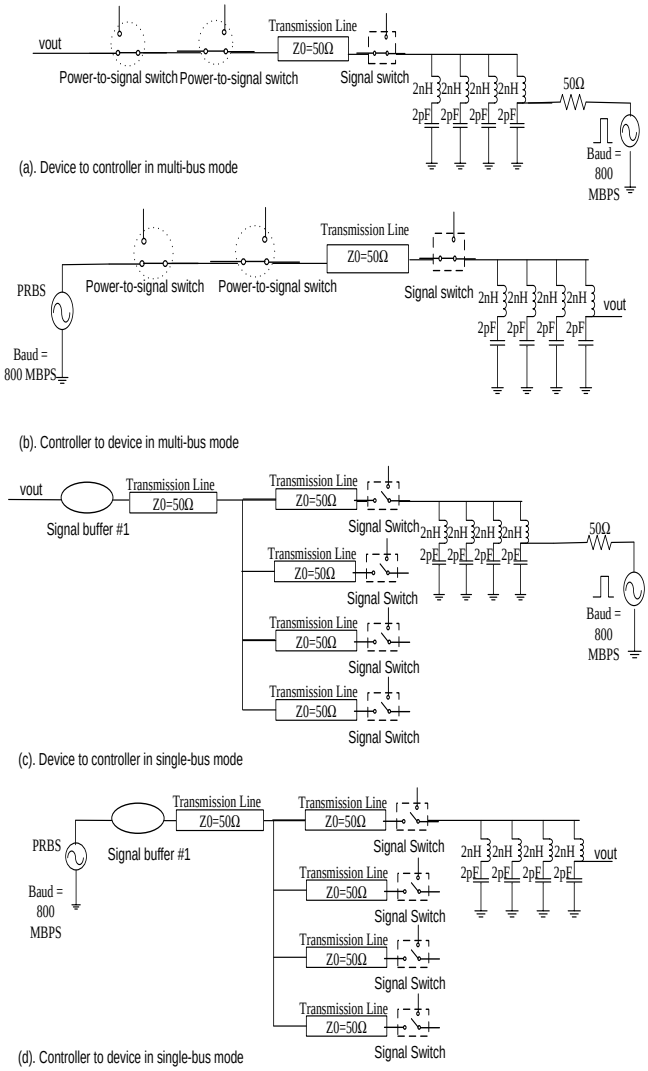


Figure 5. Spice models for signal integrity simulation

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.

3.4 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 [2] is 5,134 μm2 in 45 nm technology. This estimation is conducted by the Encounter RTL Compiler [3] with the NanGate Open Cell Library [4]. No more than 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.

3.5 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2546246, IEEE Transactions on Parallel and Distributed Systems

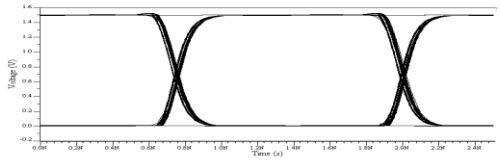


Figure 6(a). DQ Read in multi-bus mode (Device to Controller)

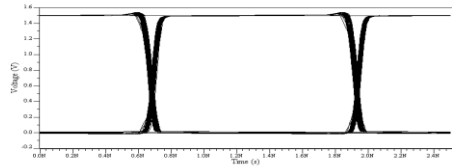


Figure 6(b). DQ Write in multi-bus mode (Controller to Device)

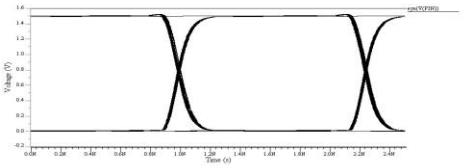


Figure 6(c). DQ Read in single-bus mode (Device to Controller)

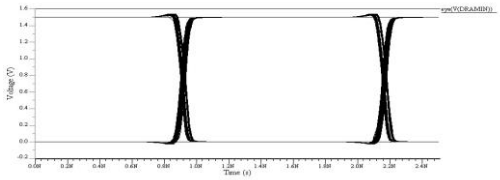


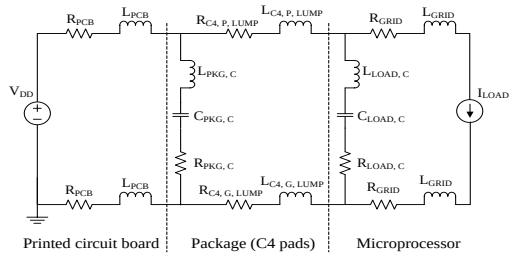
Figure 6(d). DQ write in single bus mode (Controller to Device)

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

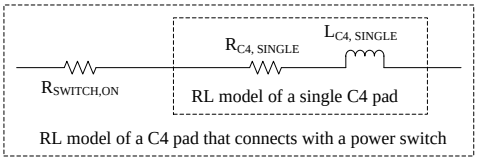
BUS	1	2	3	4
Current (A)	125	104	80	56
Voltage (V)	1	0.88	0.76	0.64
Power (W)	125	92	61	36
Frequency (GHz)	4	3.2	2.4	1.2

Table 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



(b) RL model of a C4 pad

Figure 7. RLC power delivery model

per request and extended queue delay. To reduce row-buffer conflicts, we employ XOR bank 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. We implemented XOR bank indexing which is a common way to reduce the number of row-buffer conflicts. XOR bank indexing is used in all cases including the baseline; reduces the bank access latency and the queuing delay; and helps us to show the actual benefit of our design since the queuing delay would be much larger without XOR bank indexing.

3.6 Signal Integrity

Signal integrity is analyzed to demonstrate feasibility in the sin gle-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 5. The parameters are derived from previous works [24][28]. Signal integrity challenges are alleviated since the DDR3 command signal is unidirectional and its speed is no more than that of the data signals [24]. 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 6(a-d), the eye patterns of writing data (controller to de vice) and reading data (device to controller) in the two modes are derived from the corresponding SPICE models in Figure 5(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.

3.7 Power Delivery Simulation

In this section, we assess the repercussions experienced by the PDN when the switchable pins are shifted from single-bus mode to multi-bus mode. The PDN is depicted in Figure 7(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 7(b) illustrates the RL model of the Controlled Collapse Chip Connection (C4) pads [15] in which the resistance of the on-state power switches is taken into consideration. Table 2 lists the parameter values obtained from prior work [19].

Power Delivery Network (PDN) simulations are used to calculate the maximal power consumptions and resultant frequencies as follows. 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 [21][22]. 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 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 [22], 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 3 are used in the following evaluation. For the mixed workloads, comprised of two compute-intensive programs and two memory-intensive programs, we use 2 buses and per-core DVFS [38], which is the configuration used in this experiment after we explored the configuration space for these workloads.

Wide-bus mode, introduced in Section 3.2, is used for comparing the performance of multi-bus mode. 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.

3.8 Runtime Switch Conditions

Designing a predictor to choose the most beneficial mode for the next interval is non-trivial for multi-programmed 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 ing $\widetilde{B}_{ij}(T_c) = \sum_{k=1}^{N} B_{ij}(T_c - k * T_{interval})$, where $B_{ij}(T_c - k * T_{interval})$ represents the switching benefits detailed in equation (1) and N is the length of the history to consider. The value of N should be determined based on the length of memory-intensive or computational-intensive phases in the workloads. If N is close to the length of phases, Dynamic Switching will incur a considerable overhead since it holds the system in the wrong mode for N-1 intervals. If N is chosen to be much smaller than the length of phases, Dynamic Switching become sensitive to noises and incurs some extra switching overhead. To find the optimal value of N, we tested the performance of Dynamic Switching on the selected workloads while sweeping the value of N from 1 to 4. We compared the performances of Dynamic Switching and picked up the optimal value of N, which was found to be 2 for our experiments.

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:

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

where $WS_{i,k}(T_c)$ and $WS_{j,k}(T_c)$ stand for the estimated weighted speedups for program k at time 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:

$$WS_{i,k}(T_c) = T_{alone,i,k}(T_c)/T_{shared,i,k}(T_c)$$

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

where $T_{alone,i,k}(T_c)$ stands for the execution time of the same instructions running without interfernce from co-runners and $T_{shared,i,k}(T_c)$ 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, $\text{Committed Inst}_{alone,i,k}(T_c)$ stands for the number of committed instructions during the interval following 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 $T_{alone,i,k}(T_c)$ based on the assumption that the IPS of each program is relatively steady when it runs alone, since an accurate estimation of $T_{alone,i,k}(T_c)$ is challenging [30].

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:

$$WS_{j,k}(T_c) = T_{alone,j,k}(T_c)/T_{shared,j,k}(T_c)$$

$$T_{shared,j,k}(T_c) = T_{on-core,j,k}(T_c) + T_{off-core,j,k}(T_c)$$

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:

$$T_{on-core,j,k}(T_c) = T_{on-core,i,k}(T_c) * {freq_{i,k}}/{freq_{j,k}}$$

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

$$T_{on-core,i,k}(T_c) = T_{interval} - T_{off-core,i,k}(T_c)$$

$$T_{off-core,i,k}(T_c) = T_{LLC,i,k}(T_c) + T_{DRAM,i,k}(T_c)$$

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 [32].

Table 4. 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 Parameters of DDR3-1600 from Micron datasheet [28]
PCM	tRCD=55ns, tCL=12.75ns, tRP=150ns

Table 6: 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

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:

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

$$T_{queue\ delay,j,k}(T_c) = T_{queue\ delay,i,k}(T_c) * N_{request,j,k}(T_c) \Big/ N_{request,i,k}(T_c)$$

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.

Table 5. Selected multi-programmed 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

3.9 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 [23] 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 switch less than 5 times during execution, nearly all the program phase transitions have been identified by the predictor.

4 Memory Subsystems of PCM

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

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 [1]. Phase-change memory (PCM) is a promising candidate to replace conventional memory technology to enable the continuous scaling of memory technology [26].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2546246, IEEE Transactions on Parallel and Distributed Systems

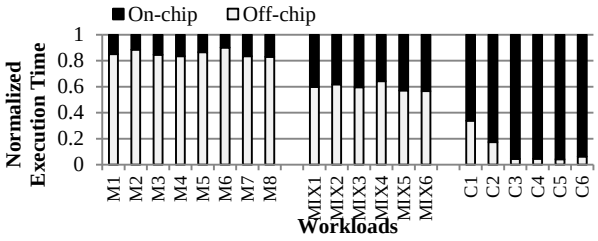


Figure 8. Normalized off-chip and on-chip latencies of workloads against total execution time.

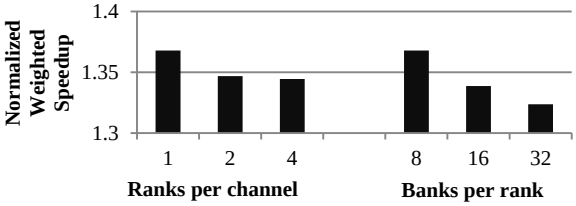


Figure 10. The average normalized weighted speedup of memory workloads in geometric mean using multi-bus mode. Each normalized against the same configuration in single-bus mode.

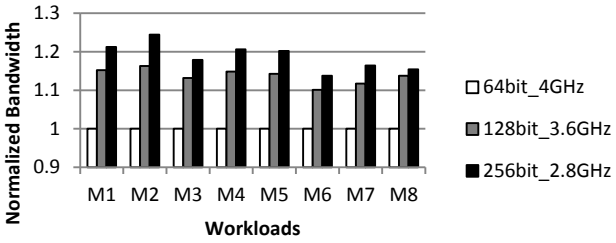


Figure 12. The off-chip bandwidth of memory intensive workloads for the baseline (core frequency of 4GHz and a memory bus of 64 bits) and two configurations of wide bus mode (core frequency of 3.6GHz and a memory bus of 128 bits, core frequency of 2.8GHz and a memory bus of 256 bits).

There are several memory subsystems proposed by architect to replace conventional memory devices using PCM devices [25][29][43]. We evaluate the benefits of switchable pins based on the performance of a PCM subsystem [26]. 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 queueing delay of memory requests. Furthermore, PCM has a 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.

5 Experimental Setup

To evaluate the benefit of our design, we simulate the x86 system documented in Table 4 using the Gem5 simulator [11]. 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 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. We also setup a timing model of PCM based on [26] shown in Table 4.

5.1 Performance and Energy Efficiency Metrics

We use weighted speedup [37] lists as follows to represent the throughput of our system shown in the following equation.

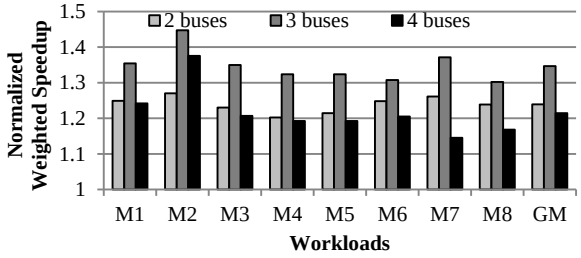


Figure 9. The normalized weighted speedup of memory-intensive workloads with 2, 3, and 4 buses against the baseline.

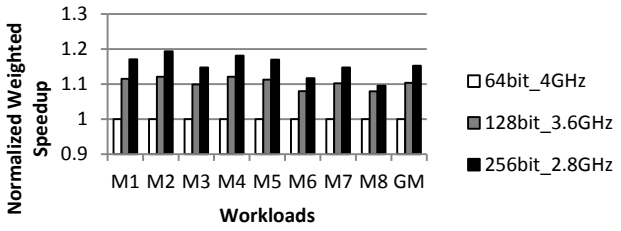


Figure 11. The performance of memory intensive workloads for the baseline (core frequency of 4GHz and a memory bus of 64 bits) and two configurations of wide bus mode (core frequency of 3.6GHz and a memory bus of 128 bits; core frequency of 2.8GHz and a memory bus of 256 bits).

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

Here, 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.

5.2 Workloads

Various multi-programmed workloads consisting of SPEC 2006 benchmarks [34] are used for our evaluation. As listed in Table 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 [7]. 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 6. 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 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 following incoming requests in the buffer.

The simulation for a mixed workload does not end until the slowest

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2546246, IEEE Transactions on Parallel and Distributed Systems

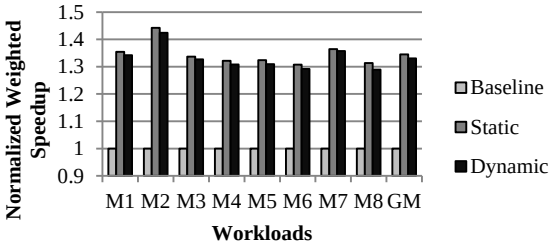


Figure 13. The normalized weighted speedup of memory intensive workloads boosted by Static Switching and Dynamic Switching with 3 buses against the baseline

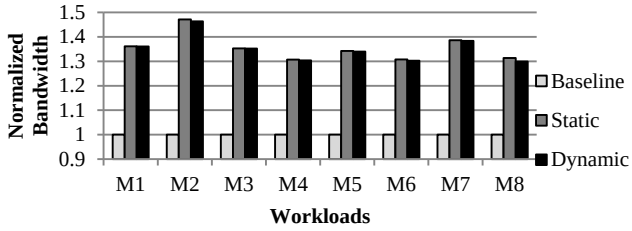


Figure 14. The normalized bandwidth of the baseline, static pin switching, and dynamic pin switching for memory-intensive workloads.

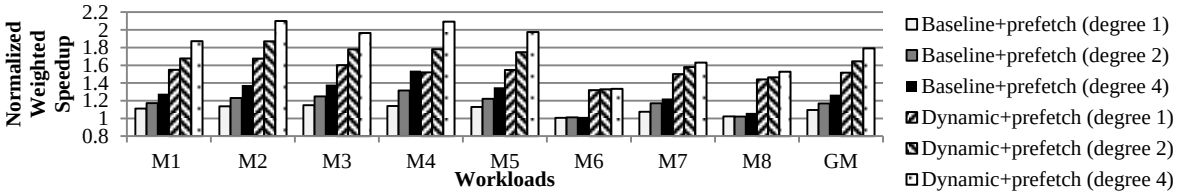


Figure 15. The improved throughput of Dynamic Switching boosted by a stride prefetcher (degree = 1, 2, 4) for memory-intensive workloads.

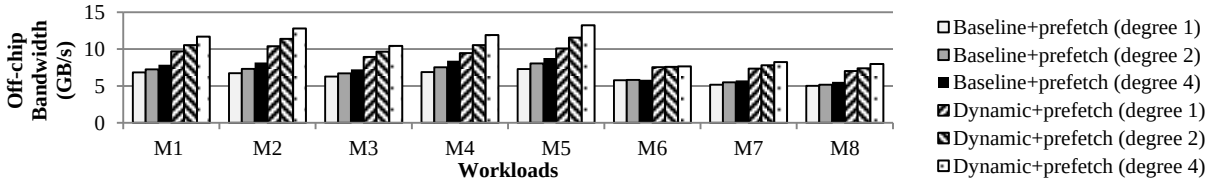


Figure 16. The off-chip bandwidth of Dynamic Switching improved by a stride prefetcher (degree = 1, 2, 4) for memory-intensive workloads.

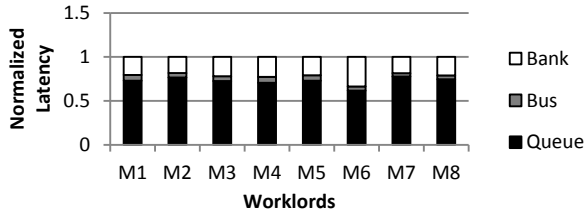


Figure 17. Breakdown of the access latency of off-chip read requests for memory-intensive workloads for the baseline.

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.

We also add four multi-threaded workloads including art [35], lbm [36], srad [12] and backprop [12] to evaluate the performance of Dynamic Switching. We manually select memory-intensive regions from the workloads and run 100 million instructions per thread in each workload.

6 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 8 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, while more than 60% of the latency of compute-intensive workloads are from on-chip latency. This implies that the memory-intensive workloads could be sped up by our Pin Switching, while the others are unlikely.

6.1 Memory-Intensive Multi-programmed Workloads

We show the performance benefits for memory–intensive workloads by switching the system into multi-bus mode. We test the system in various memory configurations and compare performance with that of wide-bus mode. Finally, we demonstrate that the benefits of multi-bus mode can be retained using Dynamic Switching and further increased by prefetching.

Figure 9 shows the performance improvements of mixed memory-intensive workloads enhanced by 2, 3, and 4 buses in multi-bus mode. The weighted speedup of each case normalized against its own baseline and the geometric mean (GM) speedup are reported here. The baseline is the simulated system fixed in the single-bus mode with 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 10 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2546246, IEEE Transactions on Parallel and Distributed Systems

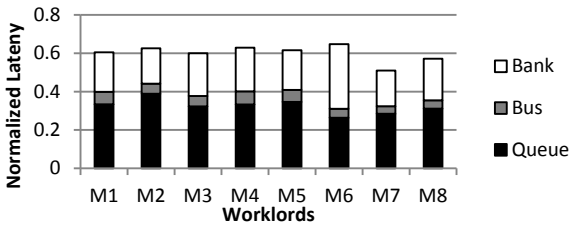


Figure 18. Breakdown of access latency of off-chip read requests for memory-intensive workloads using Dynamic Switching.

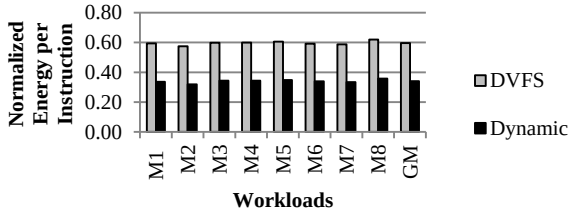


Figure 19. The normalized EPI of Dynamic Switching for memory intensive workloads with 3 buses, and the EPI from DVFS (running on 2.4GHz in single-bus mode).

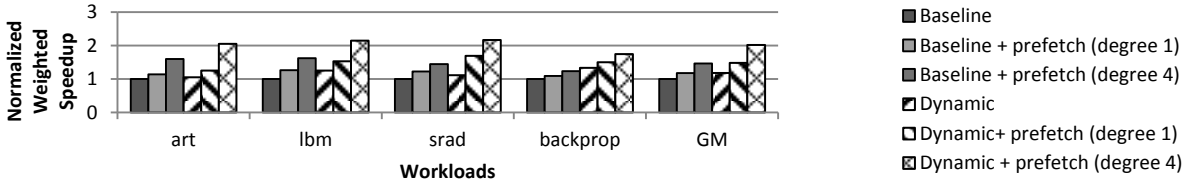


Figure 20. Performance evaluation of multi-threaded workloads with Dynamic Switching and prefetching (degree = 1, 4).

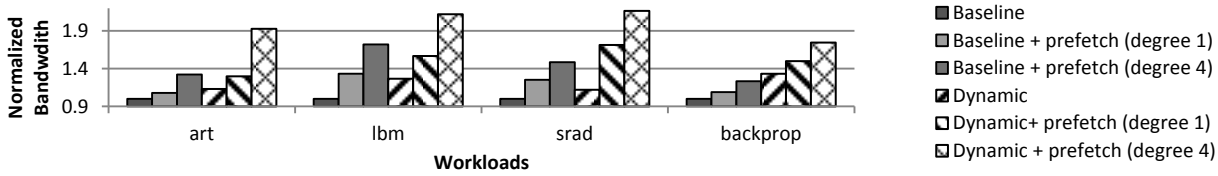


Figure 21. Normalized consumption of off-chip bandwidth of multi-threaded workloads using Dynamic Switching and prefetching (degree = 1, 4).

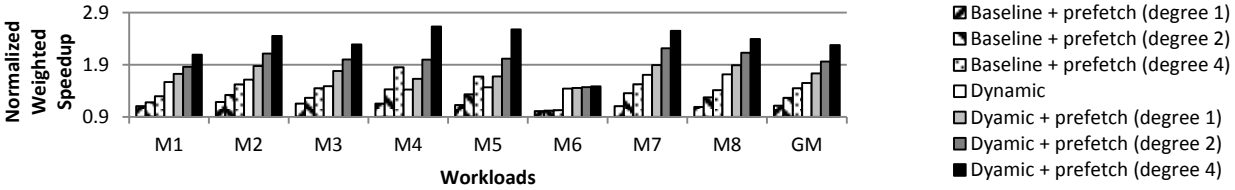


Figure 22. Improved throughput of Dynamic Switching boosted by stride prefetchers (degree = 1, 2, 4) for memory-Intensive workloads using PCM.

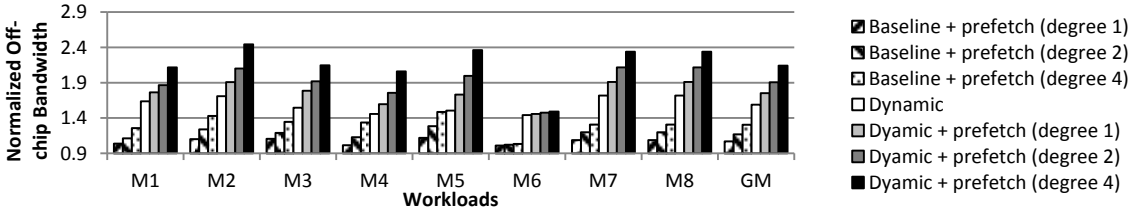


Figure 23. Normalized off-chip bandwidth of Dynamic Switching boosted by stride prefetchers (degree = 1, 2, 4) for memory-Intensive workloads using PCM.

banks per rank increases, improvement is slightly diminished due to the resulting lower row buffer hit ratio causing shorter bank access latency.

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 11 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 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 12. 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.

For multi-bus mode, we 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 in Figure 13. Both schemes are able to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2546246, IEEE Transactions on Parallel and Distributed Systems

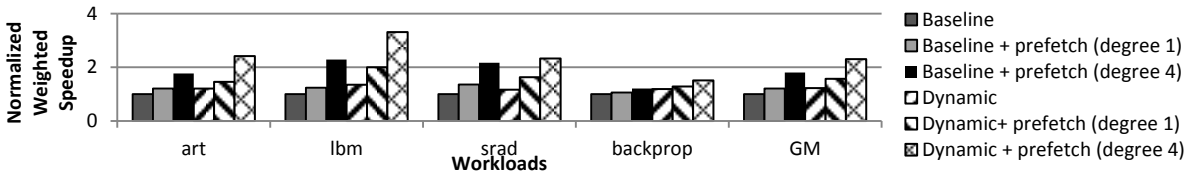


Figure 24. Performance evaluation of multi-threaded workloads using Dynamic Switching and prefetching (degree=1, 4) on the PCM subsystem.

Figure 25. Normalized consumed off-chip bandwidth of multi-threaded workloads using Dynamic Switching and prefetching (degree =1, 4) on the PCM subsystem.

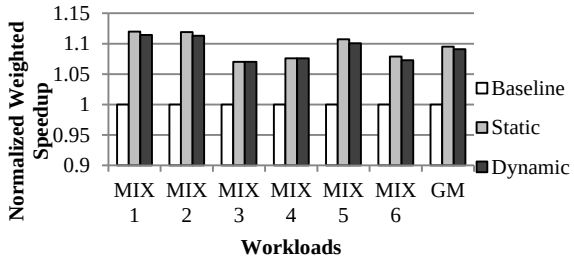


Figure 26. Normalized weighted speedup of mixed workloads boosted by Static Switching and Dynamic Switching.

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 14. The increase is similar to this of the weighted speedup in Figure 13. 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 prefetch degree of 1, 2, or 4, which denotes the number of prefetches issued on every memory reference. As illustrated in Figure 15, 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. 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 [17]. This performance improvement could be verified by the higher consumed off-chip bandwidth of Dynamic Switching shown in Figure 16. It implies that Dynamic Switching could boost the performance of the prefetcher by providing more off-chip bandwidth.

Dynamic Switching reduces the latencies of off-chip accesses by slashing the queuing delay as shown in the comparisons in Figure 17 and Figure 18. Figure 17 shows the breakdown of the access latencies for the baseline which are normalized against the total latency for each workload, while Figure 18 shows the same breakdown using Dynamic Switching which is also normalized against the total latencies of the baseline. The queueing delay in Figure 17 incurs more than 60% the total latency which is larger than the latencies of transferring data over buses and accessing banks. The approximate 40% reduction of the total latency in Figure 18 comes from the decreased queuing delay caused by Dynamic Switching. For instance, the queueing delay of the M2 workload drops from 0.76 to 0.38 while the total latency is reduced from 1.0 to 0.62.

The energy efficiency of the system could be also improved by Dynamic Switching. Figure 19 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 19 demonstrate 40% improvement in the geometric mean of the EPI for all the work loads over the baseline. Note that the overall execution time of this setting is only 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.

6.2 Memory-Intensive Multi-threaded Workloads

Figure 20 and Figure 21 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%

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2546246, IEEE Transactions on Parallel and Distributed Systems

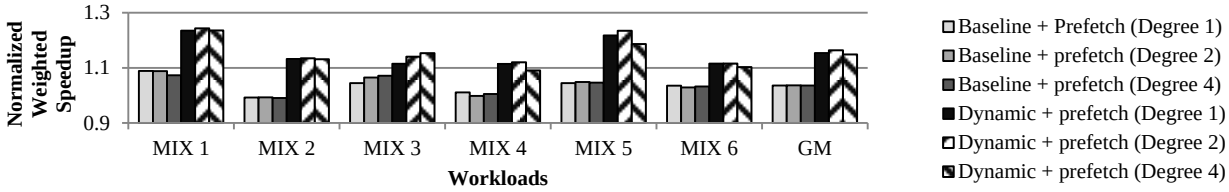


Figure 27. The improved throughput of Dynamic Switching boosted by a stride prefetchers (degree = 1, 2, 4) for mixed workloads.

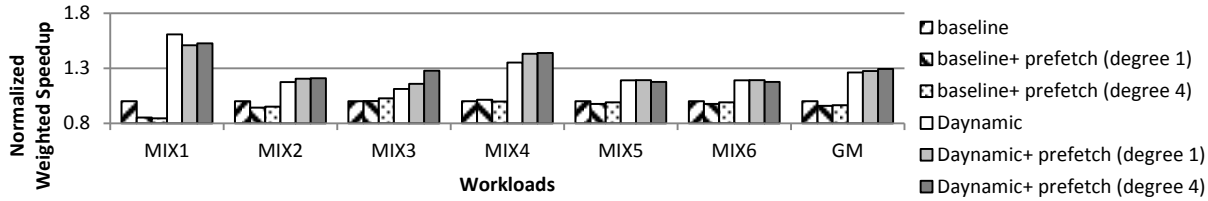


Figure 28. The improved throughput of Dynamic Switching boosted by stride prefetchers (degree = 1, 4) for mixed workloads with PCM.

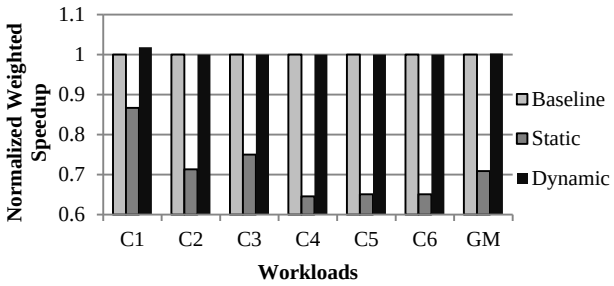


Figure 29. The normalized weighted speedup of Compute-Intensive workloads with Static Switching and Dynamic Switching.

in geometric mean providing the best performance compared to the baseline. Prefetching can exploit the benefits of multi-bus mode for multi-threaded programs, increas ing the consumed off-chip bandwidth shown in Figure 21. 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 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.

6.3 Memory-Intensive Multi-programmed Workloads on the memory subsystem using PCM

Figure 22 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 23. Dynamic switching without a prefetcher increases the off-chip bandwidth by 58% compared to the baseline, 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.

6.4 Memory-Intensive Multi-threaded Workloads on the memory subsystem using PCM

Figure 24 and Figure 25 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 25. Additionally, Dynamic Switching can mitigate the performance loss caused by the longer latency of row buffer misses in PCM. For instance, it achieves the highest performance improvement in the lbm workload which has the lowest row buffer hit rate of 57%.

6.5 Mixed Multi-programmed Workloads

Figure 26 shows the system performance improvement of mixed compute-intensive and memory-intensive workloads using Pin Switching. 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 27 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.

6.6 Mixed Multi-programmed Workloads on the memory subsystem using PCM

Figure 28 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 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%. Prefetchers 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.

6.7 Compute-Intensive Multi-programmed Workloads

Figure 29 depicts the Dynamic Switching efficiency of compute-intensive workloads in comparison to Static Switching at the cost of lower core frequency and the baseline. The geometric mean of performance degradation for compute-intensive 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 compute-intensive programs. Therefore, we do not illustrate the EPI comparison figure here.

7 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.

References

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

[2] Opencores. "DDR SDRAM Controller Core." Web. Apr. 2013. <http://opencores.org>.

[3] Cadence Design Systems, Inc. "RTL Compiler." Cadence Design Systems, Inc, Web. Apr. 2013.

[4] Silicon Integration Initiative Inc. "NanGate FreePDK45 Generic Open Cell Library." Silicon Integration Initiative, Inc,. Web. Apr. 2013.

[5] Intel Corporation, 2014a. "Desktop 4th Generation Intel® Core™ Processor Family, Desktop Intel® Pentium® Processor Family, and Desktop Intel® Celeron® Processor Family." Intel Corporation, Nov. 2014.

[6] Freescale Inc "AN3940 Datasheet." Freescale Inc., Web. Apr. 2013.

[7] Sierra Circuits. "Standard Technology PCBs." Sierra Circuits. Web. Nov. 2014.

[8] Intel Corporation. 2014b. "Pin - A Dynamic Binary Instrumentation Tool" Intel Corporation, Web. Nov. 2014,

[9] 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).

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

[11] 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. In SIGARCH Comput. Archit. News 39, 2 (August 2011), 1-7.

[12] 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 Characterization (IISWC), pp. 44-54, Oct. 2009.

[13] 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.

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

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

[16] 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.

[17] 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.

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

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

[20] 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).

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

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

[23] 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.

[24] 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.

[25] 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

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

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

[28] Micron Corp. "Datasheet of DDR3 SDRAM." Micron Corp. Web. 1 Jan. 2011.

[29] O, Mutlu. 2013. Memory scaling: A systems architecture perspective. In Memory Workshop (IMW), 2013 5th IEEE International, May 2013
[30] O, Mutlu and T. Moscibroda. 2007. Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors. In Proceedings of MICRO, 2007.
[31] O. Mutlu and T. Moscibroda. 2008. Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems. In Proceedings of ISCA, 2008.
[32] Moinuddin K. Qureshi, Daniel N. Lynch, Onur Mutlu, and Yale N. Patt. 2006. A Case for MLP-Aware Cache Replacement. In ISCA '06.
[33] Brian Rogersy, Anil Krishnaz, Gordon Bellz, Ken Vuz, Xiaowei Jiangy, Yan Solihin. 2009. Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling. In Proceedings of ISCA, 2009.
[34] Standard Performance Evaluation Corporation 2006. "SPEC CPU 2006." Standard Performance Evaluation Corporation. Web. Jan. 2013.
[35] Standard Performance Evaluation Corporation 2001. " SPEC OMP 2001." Standard Performance Evaluation Corporation. Web. Jan. 2013.
[36] J. A. Stratton, C. I. Rodrigues, I.-j. Sung, N. Obeid, L.-w. Chang, N. Anssari, G. D. Liu, and W.-m. W. Hwu.2012. Parboil : A Revised Benchmark Suite for Scientific and Commercial Throughput Computing. In the journal of Center for Reliable and High-Performance Computing, 2012
[37] Snavely and D. M. Tullsen. 2000. Symbiotic job scheduling for a simultaneous multithreading processor. In ASPLOS-9, 2000.
[38] J. Tierno, A. Rylyakov, D. Friedman, A. Chen, A. Ciesla, T. Diemoz, G. English, D. Hui, K. Jenkins, P. Muench, G. Rao, G. Smith III, M. Sperling, K. Stawiasz. 2010. A DPLL-based per core variable frequency clock generator for an eight-core POWER7 x2122 microprocessor. In VLSIC. 2010.
[39] 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.
[40] 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.
[41] 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).
[42] Runjie Zhang, Ke Wang, B.H. Meyer, M.R. Stan., K. Skadron. Architecture implications of pads as a scare resource. in Computer Architecture International Symposium. 2014.
[43] 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).
[44] 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.
[45] 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.