FlexMemory: Exploiting and Managing Abundant Off-Chip Optical Bandwidth

Ying Wang, Lei Zhang, Yinhe Han, Huawei Li, Xiaowei Li
Key Laboratory of Computer System and Architecture
Institute of Computing Technology Chinese Academy of Sciences, Beijing, P.R. China
{wangying2009, zlei, yinhes, lihuawei, lxw}@ict.ac.cn

Abstract—**The emerging nanophotonic technology can avoid the limitation of I/O pin count, and provide abundant memory bandwidth. However, current DRAM organization has mainly been optimized for a higher storage capacity and package pin utilization. The resulted data fetching mechanism is quite inefficient in performance and energy saving, and cannot effectively utilize the abundant optical bandwidth in off-chip communication. This paper inspects the opportunity brought by optical communication, and revisits the DRAM memory architecture considering the technology trend towards multi-processors. In our FlexMemory design, super-line prefetching is proposed to boost system performance and promote energy efficiency, which leverages the abundant photonic bandwidth to enlarge the effective data fetch size per memory cycle. To further preserve locality and maintain service parallelism for different workloads, page folding technique is employed to achieve adaptive data mapping in photonics-connected DRAM chips via optical wavelengths allocation. By combining both techniques, surplus off-chip bandwidth can be utilized and effectively managed adapting to the workloads intensity. Experimental results show that our FlexMemory achieves considerable improvements in performance and energy efficiency.**

Keywords-DRAM; nanophotonic; memory architecture; locality

I. INTRODUCTION

When Chip Multi-processors (CMPs) integrate more and more on-chip processing cores, they are becoming powerful enough to execute various workloads. At the same time, pin count and off-chip bandwidth are expected to grow much slower [1]. As a result, the impending "Memory Wall" is perceived as a major bottleneck to the overall system.

However, recently introduced photonic communication avoids the limitation of package pin count, making a high through-put memory interface design available [2-5]. It is announced that 10 terabyte per second bandwidth can be achieved in memory connection [5]. When the off-chip optical communication is adopted in DRAM memory, performance bottleneck is likely to shift from bandwidth limitation. Other problems like long request delay and power consumption in memory will emerge as the prime obstacles for system performance and low power design. It is estimated that nearly 30% of the total board-level power consumption in datacenter has been spent on in DRAM memory alone [6].

We will conclude later in this paper that there are mainly two factors contributing to the inefficiency of DRAM memory.

- *Rigid organization limited by I/O pins:* Traditionally, DRAM memory architecture is mainly optimized for high cell density and minimum I/O pin counts. In DRAM memory, because only a limited number of package pins can be used to carry both control commands and data signals, the 2D-arrays of storage cells are controlled in such a coarse granularity that every access command to the memory will entail reading a large data trunk from the DRAM arrays, among which only a constrained size of data bits can be placed onto memory channel. The typical design leads to sacrificing access delay and energy efficiency in memory access,
- *Locality-sensitive feature:* Current DRAMs keep continuous pages of data in the activated row-buffers, favoring the access streams with good locality. However, in the approaching multi-core era, unlike single-core applications, memory access streams in multi-core workloads exhibit much smaller degree of locality. A poor locality in data streams induces extra penalties in memory access. It will aggravate energy inefficiency and access delay issues in DRAM memory.

However, when optical connection effectively addresses the I/O pin count and bandwidth problem, it is a great opportunity to revisit the memory hierarchy and make it more efficient. We propose a novel memory design — FlexMemory. Our solution aims at improving performance and energy-efficiency in memory access by leveraging the massive optical bandwidth. Particularly, the contributions of this paper are as follows:

1) We employ super-line prefetching to fetch large data blocks from the activated row-buffers in a memory cycle via dense wavelength division multiplexing (DWDM) in optical communication, therefore exploiting a lot spatial locality in memory access.

2) We propose page folding working in coordination with super-line prefetching, which enables system to flexibly map application work sets into appropriate locations in DRAM through wavelengths allocation. With page folding activated, super-line prefetching will be adaptive to the workload characteristics by preserving locality and reducing conflicts of different access streams, which also reduces average access delay and dynamic power in row-buffers activation.

By combining the two proposed methods together, our proposed FlexMemory achieves significant performance and energy-efficiency gains under a wide range of workloads.

The remainder of the paper is organized as follows. Section II describes some insights inspired by optical communication and DRAM memory design. Section III presents the implementation of FlexMemory. In section IV, experiment method and the results are shown. Finally, section V discusses the related work, followed by the conclusion in section VI.

II. BACKGROUND AND MOTIVATION

A. Evolution of memory bandwidth : photonics technology

Photonic communication is a promising mechanism to realize data moving in low power dissipation and high

The work was supported in part by National Basic Research Program of China (973) under grant No. 2011CB302503, in part by National Natural Science Foundation of China (NSFC) under grant No.(60806014, 61076037, 60906018, 60776031, 60921002, 60831160526, 60633060).

978-3-9810801-7-9/DATE11/©2011 EDAA

throughput. There are techniques that exploit silicon photonics for on-chip and off-chip communication [2-10]. Both 3D and monolithic integration of photonic devices have been proposed in the past few years to implement processor-to-memory photonic connections [5] [9].

Nanophotonic communication techniques use light sources to generate optical carrier and waveguides to carry signals. After modulators encode data into light, the light will traverse the connected injectors and goes out through the waveguide. The silicon oxide waveguide is able to carry light of different wavelengths without interference between signals. On the receiving side, photonic detectors can absorb the light and convert the light into electrical signals. With DWDM, multiple wavelengths can share a waveguide, breaking the limitation of I/O pins and highly boosting the off-chip bandwidth.

According to recent researches on silicon photonic communication, the improvement to off-chip bandwidth brought by photonics is impressive. It is expected that we shall enter the era of tera-scale communication by substituting photonics interconnects for electrical links [11]. As illustrated in Table I, photonic communication techniques will provide more bandwidth than most computing systems can consume.

TABLE I. THE PEAKBANDWIDTH BREAKTHROUGHS IN PROCESSOR-MEMORY COMMUNICATION

	Vantrease [5]	Batten [3]	Young [11]	Gunn [9]	Kirman [4]	Stojanovic [12]
Modeled Peak bandwidth	10 TB/s	5 TB/s	>1 TB/s	10 TB/s	0.5 TB/s	1~1.5 TB/s

To prove the abundance of optical bandwidth, in our experiments, a bunch of workloads are selected for bandwidth demand estimation. It is demonstrated in Fig. 1 that the bandwidth provided by optical memory-processor connection enormously exceeds the demand of all the chosen workloads. Therefore, the observation motivates us to exploit the surplus bandwidth for system performance and energy efficiency.

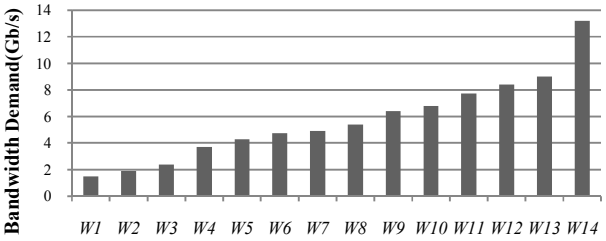

Figure 1. Bandwidth demands of sample workloads on 16-core CMP

B. Motivational analysis of inefficiency in DRAM memory

Modern DRAM memory has a strict hierarchical architecture. As shown in Fig. 2, a memory controller is connected to one or two off-chip channels. The channel is typically a wide bus that transmits memory command, data and address. Through the channel, DRAM modules can be accessed. Each memory module has multiple ranks to promote service parallelism. Because a rank is a group of DRAM devices that operates in lockstep in response to a certain command, devices 0 to N are the smallest number of chips activated by an access command. Within a rank, memory banks are the basic independent units that can be accessed in parallel. A bank spans multiple devices, being partitioned into a bunch of sub-banks. All the sub-banks in a device must share some common resources such as I/O gating that allows access to the data pins.

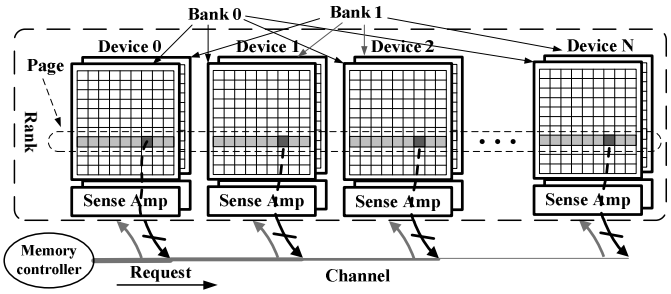

Figure 2. A generic DDRX DRAM structure with 1 channel, 1 module, 1 rank

Previously, we mentioned that the pin count limitation and locality-sensitive feature in current DRAM memory contribute to low performance and energy inefficiency. Here we take a further step to analyze the detailed sources of avoidable energy consumption and access delay in DRAM design.

1) Low Row-buffer utilization

To fetch a word-line from DRAM memory begins with first delivering the command to target channel and then activating a bank in a rank. As illustrated in Fig. 2, a single memory access entails activating all the DRAM devices in a rank and reading out a whole page of bits from the cell arrays. Among 4-64KB data latched in row-buffers, only a 64-bytes cache line will be needed to service the access request, thus the row-buffer utilization is quite low. Moreover, because read operation in DRAM cells is destructive, a write back is indispensable to recharge the activated row in the bank. The whole process is power-consuming and time-wasting.

To shorten delay and harvest energy in row activation, open-page policy is adopted to leverage spatial and temporal locality. In single-core processor, workloads typically exhibit high locality. Consequently, for conventional DRAMs, the large data trunk latched in the open row-buffers can be reused by successive requests with little interference. However, the irreversible trend of reduced locality in multi-core era makes open-page policy increasingly unacceptable. In future CMPs, it is commonplace that memory requests from different access streams compete for the limited number of channels and banks, therefore destroying the available locality. Comparatively, close-page policy is immune to penalties imposed by row-buffer miss, but completely gives up leveraging locality to amortize operation latency and energy in memory access.

2) Requests conflict

In typical DRAM design, since all the sub-banks in a device must share some common resources such as I/O gating that allows access to the data pins. Intensive access commands will compete for the memory banks, leading to conflict-induced latency and data flushing in the row-buffers. When conflict-induced stall occurs, the pending requests waiting at the memory controller are likely to increase, and it is a major factor causing the queuing delay. The conflicts can be mitigated without hurting locality if the balance between locality and service parallelism is achieved in memory.

From locality perspective, there are two solutions to raise row-buffer utilization and mitigate request conflicts at the same time. The first one is to exploit spatial locality in row-buffers as much as possible before being destroyed by interfering streams. The second one is preserve locality by avoiding interferences from access streams to the greatest extent. In our FlexMemory, we attack the problem from both angles separately and propose two techniques that leverage the abundant bandwidth.

III. FLEXMEMORY

Our baseline DRAM structure integrated with silicon photonics is shown in Fig. 3. Except the interfaces to photonics device, storage cells and arrays are organized in chips as the traditional DRAM. By attaching each independent DRAM device with photonic components, photonic signals of the modulated wavelengths will be received at the corresponding memory chips with rings in resonance. Through DWDM, memory requests carried by different wavelengths are transmitted into target memory arrays via the same waveguide and can be served in parallel, so that the communication bandwidth is greatly improved.

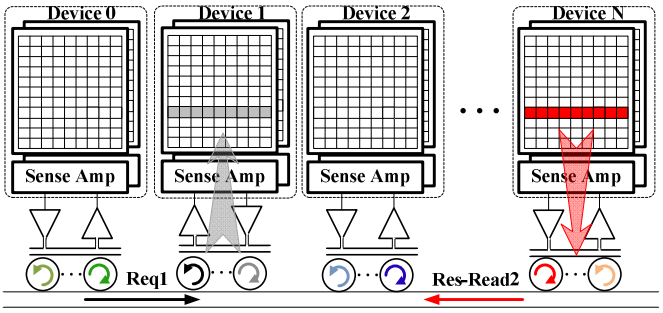

Figure 3. Optical connected FlexMemory organization

A. Bandwidth exploitation and Super-line prefetching

Through DWDM provided by integrated silicon photonics, communication bandwidth will not be scarce resources anymore. With such high concentration of photonic interconnects, massive data transfers are made easy and less costly without worrying about the transmission line effects, wiring power and heavy wiring load. We can take advantage of the abundant bandwidth to reduce cache miss latency and raise row-buffer utilization by enlarging the effective data fetch size.

Prefetching is a widely-adopted technique in memory design. There are lots of complex prefetching methods proposed to enhance performance [13-14]. However, their effectiveness is constrained by the insufficient bus width in electrical interconnects. When memory channel width is not a problem, fetching large-sized data blocks to the secondary cache will benefit a lot of applications by mitigating average cache miss latency [13]. Based on that observation, our super-line prefetching is employed to exploit spatial locality in row-buffers by utilizing optical bandwidth.

In our FlexMemory, 128 signals with different wavelengths can concurrently transmit on the waveguide, and each wavelength is modulated at 10 Gb/s [2]. Assuming all the memory devices in a channel share a common waveguide supporting 128 wavelengths and each device occupies several dedicated wavelengths, the peak bandwidth can reach more than 1000 Gb/s. That means in every memory cycle, there are enough wavelengths carrying memory data to the channel. Therefore it is feasible to fetch a larger portion of data trunks latched in the row-buffers rather than a fixed 64bytes line when a read command activates the whole row in target arrays. Compared with a 64-bytes cache line fetching, super-line prefetching can feed the bus with data trunks as large as a page size per read operation, which requires activating all the devices of different wavelengths at the same time.

For example, in Fig. 3, when each device is equipped with special ring modulators of several resonance wavelengths, the access interface of each chip will be much wider compared with DDRX DRAM in Fig. 2. If the number of wavelengths dedicated to each device is properly set, most or even all the data bits latched in the row-buffers can be placed on common waveguide per cycle. The data trunks are transmitted to secondary cache and finally consumed by processor.

Because applications have different levels of spatial locality, in our FlexMemory, super-line prefetching is adaptive to workload characteristics. The appropriate size of a "super-line" depends on the bandwidth partitioning and page-level locality of the program, which ranges from a 4KB of page size to 64byte cache line size by encoding the read command into multiple wavelengths. For example, in Fig. 3, the read2 request targeting device-N is encoded into 2 wavelengths according to the wavelengths quota of the process, so the "super-line" of 128 bytes from the activated row-buffers will be read out in response. Comparing our super-line prefetching with traditional cache line fetching, a larger portion of data read out into the row-buffers will be put on the data bus per read operation. Owning to its "reading on demand" feature, unnecessary recharging and activation activities will be greatly reduced.

B. Page folding and bandwidth management

Super line prefetching is useful in exploiting spatial locality within a program. However, in multiprogrammed workloads, if super-line fetching is enabled in multiple data streams and the prefetch size is unreasonably large, prefetching will demand excessive dedicated wavelengths for large data transfer in a memory cycle, and the resulted data bus occupancy will lead to access conflicts and queuing delay. Besides, the performance improvement gained by super line prefetching heavily depends on application characteristics, thus it is unwise to use the same prefetch size for different applications concurrently running on CMP. Conclusively, the bandwidth should be allocated and managed in order to optimize system performance, trading-off between prefetch size and service parallelism. Therefore, page folding is proposed to achieve locality and parallelism simultaneously. In brief, super-line prefetching successfully exploits the bandwidth abundance, while page folding can be employed to manage the bandwidth, benefiting from the auto-separation and auto-detection characteristics of optical signals.

To preserve locality and avoid conflicts in a wide range of applications, work sets of different tasks are adaptively mapped to different devices through page folding. For example, when there is only one task running on the processor, the data stream has a comparatively fine locality without interruptions from other tasks. In this case, more priority should be placed on request latency than throughput, and our FlexMemory system

will accommodate the data in DRAM devices as the electrical-connected DRAM does shown in Fig. 2. The physical pages of the program span the memory rows in multiple devices, so that a large super line within a physical page can be prefetched in a single memory cycle fully utilizing the 128 wavelengths dedicated to different devices. When there are multiple tasks executed in parallel on multiprocessors, the locality of each access stream is likely to be destroyed by others. As shown in Fig. 4, our FlexMemory will confine the data of different programs to separate DRAM devices so that the locality of access streams will not be disturbed by one another. In this case, the physical pages of a certain task will be "folded" into one or several dedicated chips depending on its wavelengths quota.

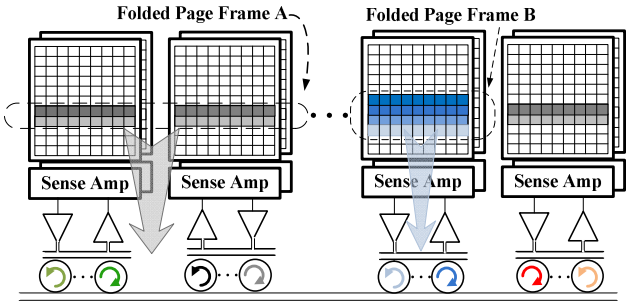

Figure 4. FlexMemory design: combining page folding and super-line prefetching. Page A and page B belong to different processes

The page folding technique is implemented via page frame allocation and wavelengths assignment. Page is the basic unit that is allocated to an application by OS. Generally the page frames corresponds to the DRAM rows which span all the devices in a rank, so the physical page addresses are directly used to index the according rows in main memory, However, as illustrated in Fig. 5, in our FlexMemory, OS-allocated page frames will be adaptively remapped to different position in memory according to the wavelengths quota of processes.

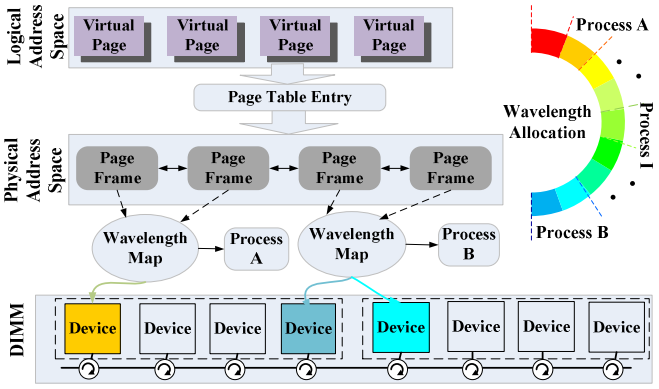

Figure 5. Page folding implementation through data layout mapping between different levels

The implementation of page folding is based on the page allocation mechanism of OS, which mostly employs buddy system to manage dynamic memory. In our FlexMemory, before being reclaimed by kernel, all the free page frames are reserved as system resource. Not until the page frames has been allocated and touched by a certain process, the data set placement in DRAM memory will not be determined. The final DRAM address command transmitted into memory modules are determined by two factors: physical page address in page table entry, the wavelengths quota assigned to the requesting process. In FlexMemory, wavelengths will be explicitly allocated to independent processes considering their page-level locality and bandwidth demands. The processes have only access to the DRAM devices equipped with ring modulators and detectors of corresponding resonance wavelengths.

Modifications should be made to process descriptor and task state segment in OS. In our system, the process descriptor, which contains all the necessary information of a single process, has a bitmap type field revealing the wavelength quota assigned by OS. Once a physical page is allocated to the thread, physical address will be filled in the page table entry. However the physical page address cannot be directly used to index the DRAM memory. It should be firstly decoded together with the according wavelength bits provided by hardware remapping table in memory controller, and the decoded address signals will be carried by special wavelengths to target memory banks. The raised hardware cost is negligible because only a limited number of entries will be contained in remapping table. Considering that the TLB is on the critical path of memory access, the wavelengths bits should be stored in it with the physical address entries. Because the wavelength bits are only needed for outstanding memory requests, page folding will cause no additional penalty to cache access.

To further exploit the bandwidth allocation, processes, which have a good page-level locality, will be allocated more wavelengths, making their physical pages span more DRAM devices, so that a large super-line in the page can be fetched per memory cycle. In extreme cases, all the data read into the row-buffers will be sent to the memory controller, completely avoiding low utilization of row-buffers and effectively hiding access latency via large prefetch size. Since bandwidth and locality profiling are not the focus of this paper, it is assumed that the programs characteristics are well studied before execution. There are also lots of available techniques to profile the bandwidth demand of workloads [15][16][17].

The extra benefits brought by page folding is that the bandwidth can be easily and accurately partitioned between different tasks, making it easy to target optimization goals like QoS, fairness or even security [17][18][19].

IV. EVALUATION AND ANALYSIS

A. Methodology

To evaluate the impacts of our FlexMemory on performance and energy efficiency, we use the full system simulator GEMS/SIMICS toolset and CACTI-6.5 DRAM modeling tool. The "memory controller" module in GEMS simulator was heavily modified to capture our FlexMemory working mechanism. Both open-page and close-page management policies with first-ready-first-come-first-serve (FR-FCFS) and batching scheduling are evaluated. The simulated system is organized as shown in Table II.

Our benchmarks are selected from SPEC CPU2000 and SPLASH-2 benchmarks. For all the programs from benchmark suites, initialization phases are skipped to ensure that only the main functionality is included. To comprehensively evaluate the performance of our FlexMemory, single-threaded, multi-

threaded and multiprogrammed workloads are simulated. For multiprogrammed workloads, various combinations of the benchmarks are used.

TABLE II. BASELINE CMP CONFIGURATION

Processor	16 core, 3.0 GHz	
Base memory Hierarchy Parameters		
L1 I cache	4-way, 64KB, 64 B-line, 1-cycle	
L1D cache	4 way, 64B-line, 32KB, LRU, 2-cycle	
L2 Cache	Shared, NUCA, 8MB, 8-way, 8 banks, LRU, 6 cycles hit	
Memory	Electrical	4G, DDR2, 4 channels, 8 Ranks, 64 Banks
	Optical	4G, 2 Ranks, 32 devices,128 wavelengths

B. Experimental Results

1) Performance Improvement

a) Single-threaded and multithreaded applications. For single-threaded and multithreaded applications, the FlexMemory is to allocate all the available wavelengths to the single program on-chip, which has no competitor for system resources. The physical pages belonging to the program will be spread across the DRAM devices, ensuring that all the wavelengths can be used for super-line prefetching. Fig. 6(a) shows the cache miss rate reduction of single-threaded and multithreaded applications for FlexMemory. Compared with the electrical baseline, there is an average 56.5% cache miss reduction for selected workloads. However, several programs gain limited performance improvement. After interpretation of application behavior, we conclude that their memory access patterns often exhibit burst characteristics, leading to the successive cache misses in a short interval. In this case, super-line prefetching cannot dramatically reduce the penalty caused by long memory access latency.

b) Multiprogrammed workloads We have been emphasizing on the adaptiveness of FlexMemory, which can flexibly balance between bandwidth demand and parallelism requirement. This feature is manifested in the scalable performance for various workloads. For multiprogrammed workloads, we use different combinations of SPEC CPU2000 and SPLASH-2 benchmarks. Fig. 6(b) shows the speedup of FlexMemory model for various workload mixes. The chosen mixes encompass 15 applications from SPLASH-2 and SPEC CPU2000 suites. From mix-1 to mix-14, the workloads become more and more memory-intensive because the number of programs concurrently running on CMP keeps increasing. As with Fig. 6(b), the lightest workload mix-1 is composed of only two programs while mix-14 contains 15 programs, each of which is bound onto one processing core. In experiments, the wavelengths allocation is based on the simulation result profiled in pre-execution phase.

As shown in Fig. 6(b), most workloads have more than 1.3x speedup. However, for mix-13 and mix-14, they have gained slight performance improvements because the cache capacity becomes a bottleneck. When the CMP is loaded with heavy workloads that prefetch in a coarse granularity, conflict misses in cache increase quickly.

2) Rowbuffer utilization

Because bitline charging activities in row-buffers contribute to most of the dynamic DRAM power [20], we choose row-buffer utilization (RBU) as the major metric to evaluate the energy efficiency in memory operation. RBU indicates the percentage of the row-buffer data finally consumed by the processors during the whole execution time, which depends on three factors: the row activation times, the activated row size and total memory data size touched by processing cores (over-fetched data not included). All the results are normalized to the baseline DDR2 DRAM with close-page policy.

a) Single-threaded and multithreaded applications. Fig. 7(a) plots the row-buffer utilization for single programs. For our FlexMemory, through page folding (PF) and super-line prefetching (SP), data bits in row-buffers spanning 32 devices are prefetched and consumed, ensuring a much higher utilization than the baseline DRAM model.

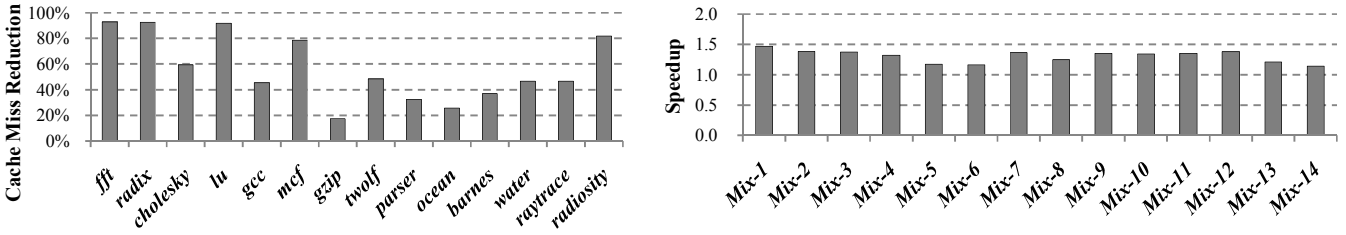

Figure 6. (a) Cache miss reduction of single-threaded and multithreaded programs. (b) Performance improvement of multiprogrammed applications

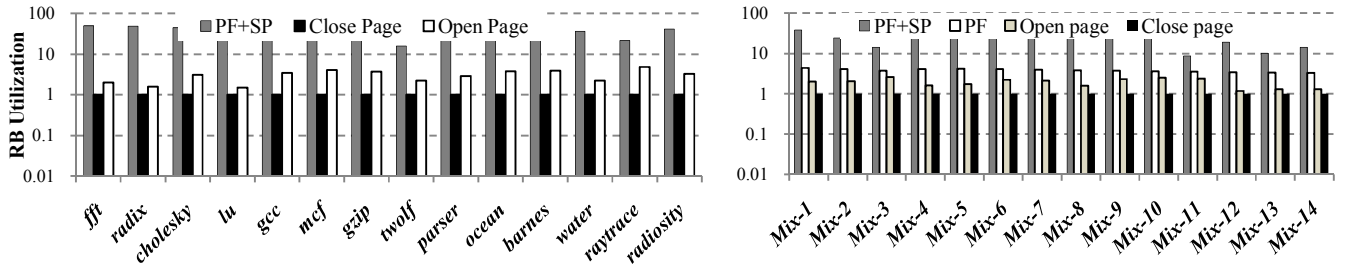

Figure 7. (a) RBU of single-threaded and multithreaded programs. (b) RBU of multiprogrammed programs. PF+SP means FlexMemory working in paging folding and super-line prefetching mode, employing close-page.PF means FlexMemory without super-line prefetching, in this mode, employing open-page.

b) Multiprogrammed workloads. For multiprogrammed workloads, the evaluated RBU is reported in Fig. 7(b). Compared to the open-page baseline, the close-page baseline is clearly worse in terms of row-buffer utilization because every access should reactivate the entire row. However, when the number of concurrent programs increase, the normalized RBU of open-page baseline is continuously shrinking because of a dramatic drop in the row-buffer hit rate. For our FlexMemory combining page folding and super-line prefetching, RBU is greatly promoted. The explanation is obvious. Adaptive page folding can effectively improves the spatial locality of successive access through data set isolation, and super-line prefetching put larger fractions of row-buffer data onto the optical channels sharing a single waveguide.

V. RELATED WORK

a) Memory Design. There are techniques such as mini-rank [21] and multicore DIMM [22] aimed at lowering DRAM operation power by reducing effective row size in DRAM modules. A. Udipi et al. propose SBA and SSA to decrease the dynamic power in memory access [23]. "Micro-page" is proposed to attack energy inefficiency problem from OS perspective [20]. These approaches save energy by reducing the ratio of activated bits in DRAM chips, and often result in performance degradation. In addition, they could not avoid the limitation of electrical connection, which prevents major changes to the DRAM organization. Other research work on DRAM like 3-D architectures is also described [24].

b) Photonic communication. There are techniques that exploit photonics for on-chip and off-chip communication [2-10]. Batten et al. describe a low power processor-memory connection for future manycore systems [3]. Krishnamoorthy et al. examine future opportunities for adopting photonic communication into a high-performance computing system at different levels [7]. Vantrease et al. propose Corona to use nanophotonic interconnect for both inter-core and off-stack communication [5]. Beamer et al. use DWMM to reduce intra-chip link power in DRAM memory communication [2].

Unlike other traffic-avoiding techniques or single-objective solutions, our FlexMemory attacks problems in DRAM from a different angle by utilizing the surplus photonic bandwidth.

VI. CONCLUSION

Emerging nanophotonic technology greatly promotes bandwidth in memory communication. Based on the observation, we review the inefficient architecture of conventional DRAM memory and propose a novel DRAM memory design, which effectively utilizes the optical bandwidth resources. The proposed FlexMemory avoids the conflict-inducing and energy-inefficient access mechanism in traditional DRAM memory. Combining page folding and super-line prefetching technique together, our FlexMemory can preserve and exploit locality in different applications. In evaluation of various workloads, we found that our design improves performance and row-buffer utilization significantly.

REFERENCES

[1] ITRS. International Technology Roadmap for Semiconductors, 2007 Edition. http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[2] Scott. Beamer, Chen. Sun, Yong-jin. Kwon, Ajay. Joshi, Christopher Batten, Vladimir. Stojanovic, Krste. Asanovic, "Re-Architecting DRAM Memory Systems with Monolithically Integrated Silicon Photonics", International Symposium on Computer Architecture, 2010, pp.129-140.

[3] Batten. C, Joshi. A, Orcutt. J, Khilo. A, Moss. B, Holzwarth. C. Popovic, M., Hanqing Li, Smith. H, Hoyt. J, Kartner. F, Ram. R, Stojanovic. V, Asanovic. K, ''Building Manycore Processor-to-DRAM Networks with Monolithic Silicon Photonics,'' HOTI 08, 2008, pp. 21-30.

[4] Kirman. N. Kirman, M. Dokania, R.K. Martinez, J.F. Apsel, A.B. Watkins, M.A. Albonesi. D.H, ''On-Chip Optical Technology in Future Bus-Based Multicore Designs,'' Micro, IEEE Volume: 27. Issue:1, 2007 , pp 56 - 66.

[5] Vantrease. D, Schreiber. R, Monchiero. M, McLaren. M, Jouppi. N.P, Fiorentino. M, Davis. A, Binkert. N, Beausoleil. R.G, Ahn. J.H, ''Corona: System Implications of Emerging Nanophotonic Technology,'' ISCA 08 , IEEE CS Press, 2008, pp. 153-164.

[6] L. Barroso and U. Holzle. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Morgan& Claypool, 2009.

[7] A.V. Krishnamoorthy, Lexau. J, Xuezhe Zheng; Cunningham. J.E; Ho. R., Torudbakken. O, ''Optical Interconnects for Presentand Future High-Performance Computing Systems,'' Proc. Hot Interconnects (HOTI 08), IEEE CS Press, 2008, pp. 175-177.

[8] Binzhang Fu, Yinhe Han, Huawei Li, Xiaowei Li, "Accelerating Lightpath setup via broadcasting in binary-tree waveguide in Optical NoCs Design," Automation & Test in Europe Conference & Exhibition (DATE), 2010, pp: 933 - 936

[9] C. Gunn," CMOS photonics for high-speed interconnects," IEEE Micro, 26(2), Mar. Apr. 2006, pp.58–66.

[10] Briere. M, Girodias. B, Bouchebaba. Y, Nicolescu. G, Mieyeville. F, Gaffiot. F, O'Connor. I, "System Level Assessment of an Optical NoC in an MPSoC Platform," Design, Automation & Test in Europe Conference & Exhibition, 2007, pp: 1 - 6

[11] Young. I.A, Mohammed. E, Liao. J.T.S, Kern. A.M, Palermo. S, Block. B.A, Reshotko. M.R, Chang. P.L.D, "Optical I/O Technology for Tera-Scale Computing," Solid-State Circuits. IEEE Journal of Volume: 45, 2010 , pp: 235 - 248

[12] Stojanovic. V, Joshi. A, Batten. C, Yong-Jin Kwon; Beamer. S, Sun Chen; Asanovic, K, "CMOS photonic processor-memory networks," Photonics Society Winter Topicals Meeting Series (WTM), 2010, pp: 118 - 119

[13] O. Temam, Y. Jegou, "Using Virtual Lines to Enhance Locality Exploitation," In Proceedings of the International Conference on Supercomputing, 1994.

[14] W. Lin, S. Reinhardt, D. Burger, "Reducing DRAM Latencies with an Integrated Memory Hierarchy Design," In Proceedings of the International Symposium on High Performance ComputerArchitecture, 2001.

[15] Ebrahimi, E., C. J. Lee, Onur. Mutlu, Yale. N. Patt, "Fairness via Source Throttling: A Configurable and High-Performance Fairness Substrate for Multi-Core Memory Systems," ASPLOS. 2010

[16] N. Rafique, W. Lim, and M. Thottethodi, "Effective Management of DRAM Bandwidth in Multicore Processors," in Proc. of the 16th International Conference on Parallel Architectures and Compilation Techniques(PACT), 2007.

[17] Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," in Proc. of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2007.

[18] Liu. F, Understanding How Off-Chip Memory Bandwidth Partitioning in Chip Multiprocessors Affects System Performance. HPCA.2010

[19] K. Nesbit, D. Aggarwal, J. Laudon, and J. Smith, "Fair Queuing Memory System," MICRO-39. 39th Annual IEEE/ACM International Symposium on, 2006, pp.208 – 222

[20] K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramonian, A. Davis, "Micro-Pages: Increasing DRAM Efficiency with Locality-Aware Data Placement," In Proceedings of ASPLOS-XV, 2010.

[21] H. Zheng, Jiang Lin; Zhao Zhang, Gorbatov. E, David. H, Zhichun Zhu; "Mini-Rank: Adaptive DRAMArchitecture For Improving Memory Power Efficiency," In Proceedings of MICRO, 2008.

[22] J. Ahn, Leverich. J, Schreiber. R.S, Jouppi. N.P, "Multicore DIMM: An energy-efficient memory module with independently controlled DRAMs,"EEE Computer Architecture Letters, 8(1), 2009, pp.5–8

[23] Aniruddha Udipi, Naveen Muralimanohar, Niladrish Chatterjee, Rajeev Balasubramonian, Al Davis, Norm Jouppi, "Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores," ISCA. 2010

[24] Loh, G. H. (2008). "3D-Stacked Memory Architectures for Multi-core Processors," 35th ISCA, 2008, pp.453 - 464