A 2.5-D Memory-Logic Integration With Data-Pattern-Aware Memory Controller

Dongjun Xu
Xi'an University of Technology and Nanyang Technological University

Sai Manoj P. D., Kanwen Wang, and Hao Yu
Nanyang Technological University

Ningmei Yu
Xi'an University of Technology

Mingbin Yu
A*-STAR Institute of Microelectronics (IME)

Editor's notes:
This paper presents silicon interposer-based 2.5-D integration of core and memory chips. Utilization of the channels through TSVs and interposer routing between the core and memory chips is maximized by bandwidth balancing enabled by space-time multiplexing of the channels with core clustering.

—Sung Kyu Lim, Georgia Institute of Technology

THE FUTURE DATA-ORIENTED computing systems involve huge communication bandwidth demand from cores when accessing the shared main memory, which has fundamentally limited bandwidth by 2-D integration [1]–[3]. The 3-D integration by through silicon via (TSV) [1], [4] can significantly improve memory-logic input/output (I/O) communication bandwidth. However, the 3-D integration by TSVs has poor heat removal capability. As the currently shared main memory is usually DRAM with significant leakage in advanced nodes, it is quite susceptible to thermal runaway when a positive feedback is formed due to electrical–thermal coupling in the 3-D integration [2], [3]. Recent development of 2.5-D integration by through silicon interposer (TSI) [5] is one promising solution by integrating multiple dies on one common substrate. It has shown good thermal dissipation as well as high bandwidth and low power when realized as transmission line (T-line) underneath the substrate. In this paper, we have compared the power and thermal characteristics for memory-logic integrated many-core systems by 3-D TSV and 2.5-D TSI under various data-oriented benchmarks, and have found that the 2.5-D integration is more thermally resilient than 3-D integration with very low possibility for thermal runaway failure to happen.

However, a 2.5-D integration by TSI I/Os has less bandwidth utilization compared to 3-D integration by TSV I/Os. As there are limited TSI I/O channels to access the shared memory, this calls for a channel reutilization under quality-of-service (QoS) constraint, which involves memory controller management [6]–[8] because it is commonly employed as the bridge between cores and shared memory. The management of memory controllers can be classified into static and dynamic upon whether the requests are

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.
Digital Object Identifier 10.1109/MDAT.2015.2440413
Date of publication: 00 Month 0000.

Copublished by the IEEE CEDA, IEEE CASS, IEEE SSCS, and TTTC

determined at runtime or design time. Major drawback of the static memory controller [7] is the limited capability only for one already specified application at design time, thus limiting the utilization rate. The dynamic memory controller [8] can perform scheduling at runtime based on the real-time requests, but nonscalable yet for the many-core microprocessor.

In this paper, based on the data pattern in terms of memory-access characteristics, we propose a reconfigurable dynamic memory controller for the reuse of 2.5-D I/O channels such that one can improve the channel utilization with good QoS for the many-core microprocessor. To satisfy the demand from cores to access one shared memory with matched supply of 2.5-D I/O channels, reconfiguration of I/O channel connection is performed by space-time multiplexing using crossbar switches. Different application workloads are first characterized based on the magnitude of memory accesses. Accordingly, the I/O channels are allocated to different clusters, called space multiplexing. Moreover, inside one cluster, cores will be assigned with different priority that is further characterized by the phase of memory accesses to occupy the channels, called time multiplexing. Such a space-time multiplexing improves the effective utilization of available 2.5-D I/O channels, hence resulting in a balanced bandwidth with improved QoS.

The remainder of the paper is organized as follows. The Comparison of 3-D and 2.5-D many-core microprocessors section presents the comparison of 3-D and 2.5-D integrations. The 2.5-D architecture of the many-core microprocessors with shared memory section describes the 2.5-D many-core system architecture with formulated problem of space-time multiplexing in the memory controller design. The Memory-access data-pattern section presents the analysis of the memory-access data patterns. The Reconfigurable memory controller with space-time multiplexing section presents the space-time multiplexing algorithm for 2.5-D I/O management. Then, we discuss experimental results and conclusions.

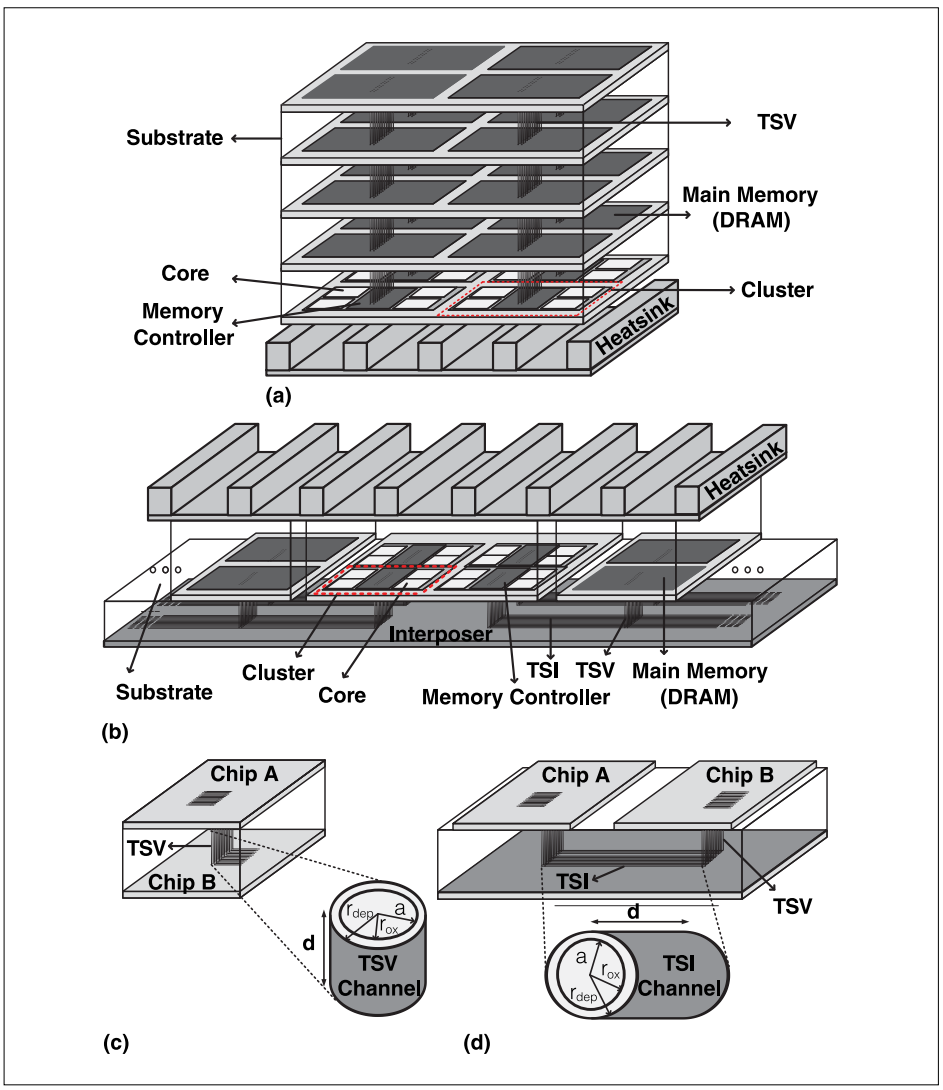


Figure 1. (a) Three-dimensional integration by TSV I/O. (b) Two-and-a-half-dimensional integration by TSI I/O. (c) Three-dimensional TSV I/O structure. (d) Two-and-a-half-dimensional TSI I/O structure.

Comparison of 3-D and 2.5-D many-core microprocessors

3-D and 2.5-D integrations

Figure 1 shows memory-logic integrated many-core microprocessors by 3-D TSV and 2.5-D TSI, respectively. As shown in Figure 1a and c, a face-to-back

bonding is applied between two layers: the many-core layer and the DRAM layer, using TSVs, which are modeled as short-distance resistance–capacitance (RC) interconnect. The heat-sink is at the bottom with poor heat-dissipation path when using 3-D stacking. DRAM has significant leakage power, which can be exponentially amplified with temperature increase when environmental heat cannot be effectively removed. As such, there is severe thermal reliability concern when a positive feedback is formulated under the electrical–thermal coupling. In contrast, as shown in Figure 1b and d, for 2.5-D integration with TSIs, the substrate is drilled with TSIs formed underneath to connect two dies. The TSIs can be modeled as middle-distance transmission lines (T-lines). As memory banks and cores can be spread uniformly on one common substrate that is close to the heat sink, it has a much better heat dissipation path with much less thermal reliability concern.

Thermal runaway in 3-D

The many-core system architecture shown in Figure 1 is implemented in Gem5, a cycle-accurate performance simulator. Application benchmarks such as SPEC 2006 and Phoenix are utilized to provide a comprehensive measure of performance across wide range of workloads. McPAT and CACTI are used to provide the delay, power, and area of the core and memory blocks. TSV length of 50 μm and TSI length of 1.5 mm are chosen for similar integration capacity. DRAM with four banks, each of 64 MB, occupies an area of 32.025 mm^2. With the power traces, a thermal simulator Hotspot is employed to provide temperature profile considering electrical–thermal coupling due to temperature-dependent leakage current. The heat-sink size is set as 4.0 cm $\times$ 4.0 cm in 2.5-D and 2.0 cm $\times$ 2.0 cm in 3-D, both with a heat-removal resistance as 4.6 K/W. The parasitics of TSV and TSI are considered similar to [3].

Figure 2a shows the system power breakdown for 3-D and 2.5-D integrations, respectively. Power consumption for 2.5-D integration executing 401.bzip2 and *K*-means benchmarks at 25 °C and 120 °C is shown in Figure 2a(i) and (ii), respectively. Similar power breakdown for 3-D integration is shown in Figure 2a(iii) and (iv), respectively. One can observe that the power of DRAM has become dominant when temperature is increased due to electrical–thermal coupling through leakage current.

Moreover, we vary the number of stacked memory layers for 3-D integration, i.e., three layers means it has one logic layer and two memory layers. The initial temperature is at the room temperature (25 °C). The dynamic system temperature trend is shown in Figure 2b for both 2.5-D integration and different sets of 3-D integrations. When the temperature goes beyond a threshold (100 °C), meaning that the heat sink cannot dissipate the heat produced by the system, there can exist a positive feedback, resulting in dramatic increase of system temperature, called thermal runaway failure. One can find that the temperature profile of 2.5-D integration can remain in safe region all the time. But for a five-layered 3-D integration, the dramatic temperature rising (or thermal runaway) can be observed after six million execution cycles.

As such, for the sake of thermal reliability, one would choose the 2.5-D integration for the many-core microprocessor. However, the bandwidth of the many-core microprocessor is limited when compared to the 3-D integration. As such, one needs to further develop an efficient I/O channel utilization in the 2.5-D integration.

Two-and-a-half-dimensional architecture of many-core microprocessor with shared memory

As there exist time-varying bandwidth demands from cores to access the shared memory, TSI I/O channels may not be fully utilized under a fixed allocation. In this section, we present a problem of 2.5-D I/O channel reutilization by the I/O management in the memory controller, which is the bridge between many cores and shared memory. A space-time multiplexing of 2.5-D I/O channels is introduced to achieve the demand–supply matching with the constrained communication QoS.

Two-and-a-half-dimensional integration with reconfigurable memory controller

Figure 3a shows an overview of the many-core microprocessor and the shared memory by the 2.5-D integration with TSIs. Microbumps are used to connect the core, memory, and TSIs. Figure 3b shows the top view of 2.5-D system architecture. It consists of two different dies (indicated by dotted lines) on one common substrate connected by the TSI I/Os. One die is composed of the many-core microprocessor and the other die has the shared main memory. Core accesses the main memory through a crossbar switch network [9], configuring TSI I/O channel

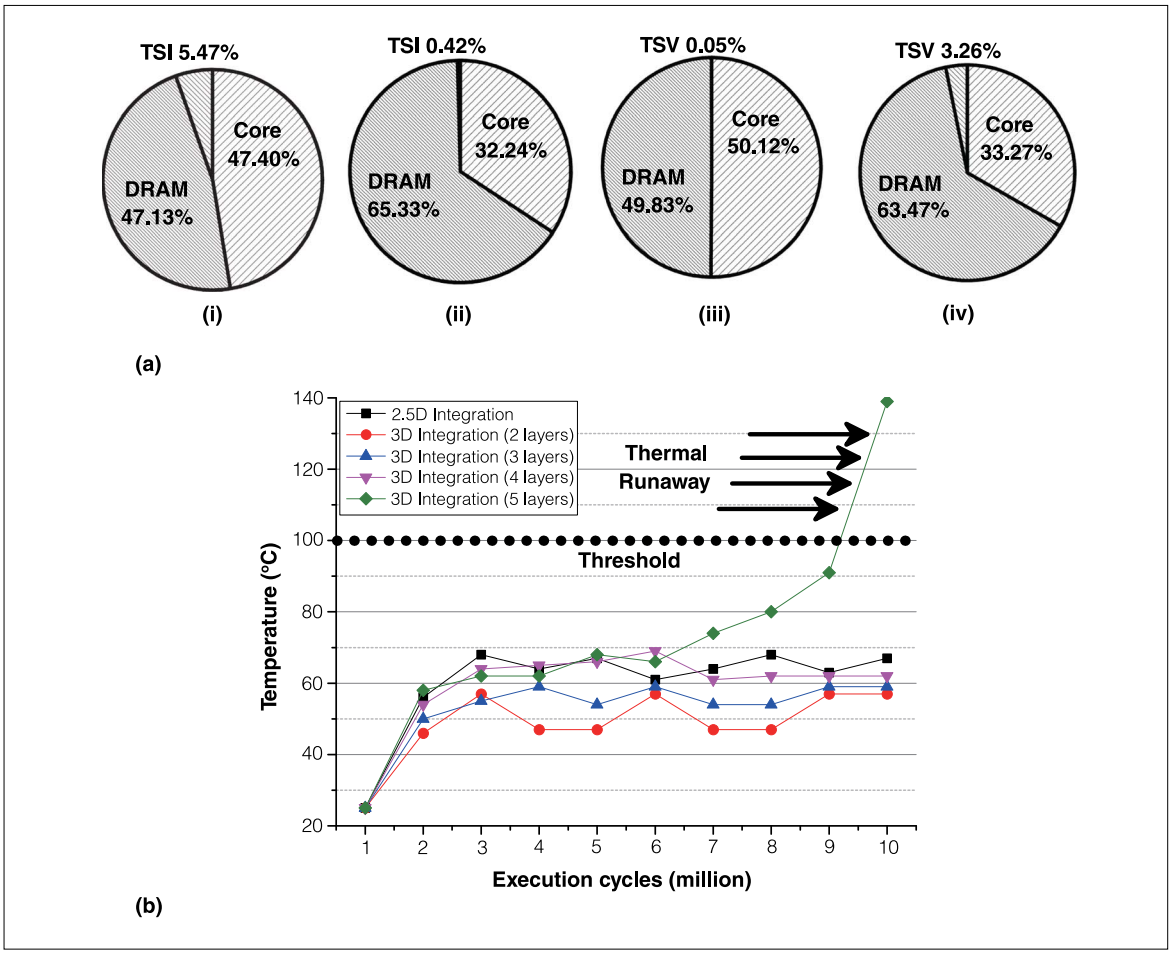


Figure 2. (a) Power breakdown of (i) 2.5-D integration at 25 °C; (ii) 2.5-D integration at 120 °C; (iii) 3-D integration at 25 °C; and (iv) 3-D integration at 120 °C. (b) Thermal runaway analysis by 3-D and 2.5-D integrations.

connections between the cores and the shared main memory. The crossbar switch network is suitable here with the following advantages: simple one-hop routing and ease of implementing QoS policy. Besides, it can be easily reconfigured based on the data-pattern analysis, discussed later. Note that the switch-network can be dynamically reconfigured to connect with TSI I/O channels. The TSI I/O channel management in a memory controller can be implemented by simple logics of the address decoder, the data queue, the request queue, and the scheduler.

Problem formulation

To manage I/O channels with time-varying bandwidth demands from many cores when accessing the shared memory, the main idea here is to learn the memory-access data patterns such that one can perform a reconfigurable space-time multiplexing inside the memory controller to fully utilize the TSI I/O channels. As such, the complexity for a large-scale I/O management can be reduced to implement inside the memory controller. Indeed, one can formulate a space-time multiplexing problem with demand–supply matching as follows.

Problem. A reconfigurable memory controller can perform space-time multiplexing that adaptively allocates N_{ch} TSI I/Os to N_c cores such that

$$\begin{aligned} \min: \quad & \sum_{i=1}^{N_c} |B_a(c_i) - B_d(c_i)| \\ \text{s.t.}: \quad & \text{(i) } B_a(c_i) \geq B_d(c_i) \\ & \text{(ii) } Q_i \leq E_R \qquad \forall i = 1, 2, \ldots, N_{ch} \end{aligned} \quad (1)$$

where $B_d(c_i)$ and $B_a(c_i)$ are the demanded and allocated bandwidths for core c_i, respectively; Q_i

and E_R are the number of requests at ith channel and the maximum request capacity of one channel, respectively.

Memory-access data pattern

In this section, we present memory-access data-pattern analysis with the according QoS analysis that can be utilized for space-time multiplexing. We define the control cycle with period T_c and each control cycle is further divided into time slots with period T_s. Within each control cycle T_c, I/O channels are allocated, while at each time slot T_s, cores are allocated with I/O channels.

LLC MPKI pattern

There exist many approaches to describe the memory-access data pattern of workloads. Last-level cache (LLC) misses-per-kilo instructions (MPKIs) are an important metric to indicate the communication intensiveness between cores and memory. Higher LLC MPKI indicates larger bandwidth requirement. Since this paper focuses on the memory-logic communication, LLC MPKI is considered for analyzing the memory-access data pattern. Note that the memory-access number can be inferred by LLC MPKI [10]. Memory-access number $M(c_i)$ for core c_i within control cycle T_c is the sum of the access number at each time slot T_s of T_c, given as

$$M(c_i)\big|_{T_c} = \sum_{t=1}^{\frac{T_c}{T_s}} M(c_i)\big|_t. \quad (2)$$

The bandwidth demand $B_d(c_i)$ of core c_i is related to the memory-access number $M(c_i)$ by

$$B_d(c_i) = \frac{M(c_i) * L_c}{T_c} \quad (3)$$

where L_c is the LLC line size.

Based on the memory-access number, one can classify the communication traffic pattern of cores. Cores with similar bandwidth demands can further vary in arrival time of memory-access requests. As such, one can prioritize the cores based on the request arrival time to allocate the channel.

Quality of service

The system performance is sensitive to long-latency memory requests because instructions dependent on the long latency load cannot proceed until the load completes. More memory access served indicates a higher QoS and better performance. Here, QoS is considered as the metric to determine the demand–supply matching, given by

$$\text{QoS} = \frac{\sum_{i=1}^{N_c} r_i M(c_i)}{\sum_{i=1}^{N_c} M(c_i)} \quad (4)$$

where r_i is the processed ratio of requests for core c_i.

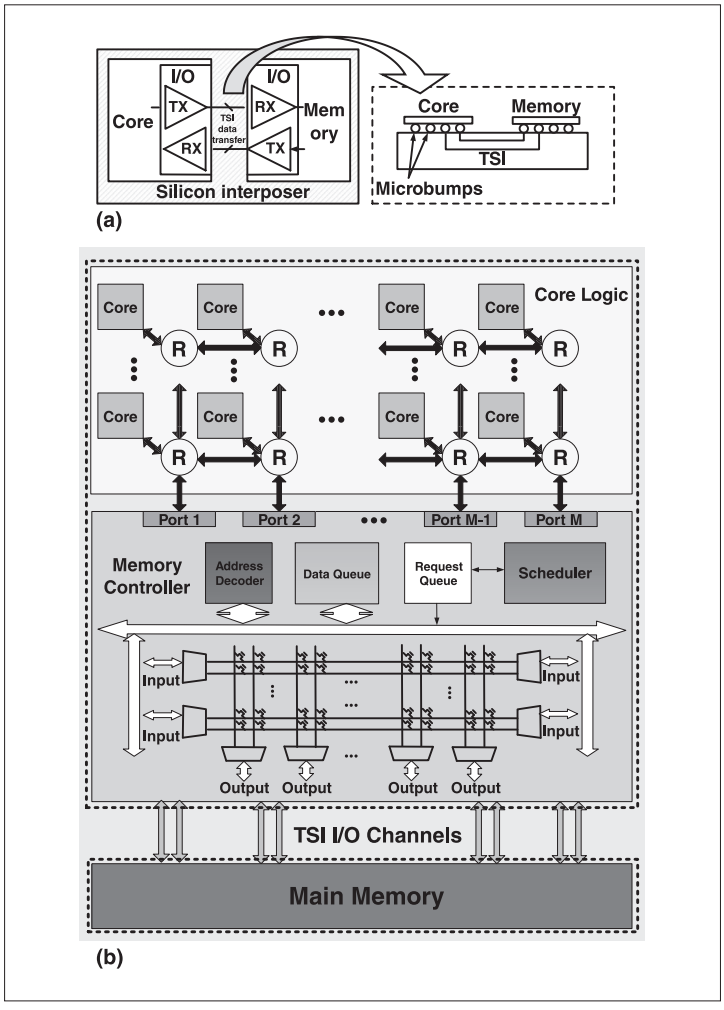


Figure 3. (a) Two-and-a-half-dimensional memory-logic integration architecture. (b) Top view of the many-core microprocessor and main memory integration by 2.5-D I/O channels.

Reconfigurable memory controller with space-time multiplexing

To solve the demand–supply matching problem, a space-time multiplexing based on the memory-access data pattern is proposed here, which can be implemented inside the memory controller.

Space multiplexing: Channel allocation

First, we discuss the adaptive allocation of I/O channels to cores. The demand here is the bandwidth requirement from the cores running workloads, and supply is the I/O channels to access shared main memory. To deal with a large number of cores, we cluster the cores based on the magnitude of the memory-access data pattern (LLC MPKI).

Clustering is defined as the process of grouping cores with similar bandwidth demands. Each cluster is allocated with a different number of cores and connected to one of the M ports of the memory controller through on-chip routers. Inside the memory controller, cores are virtually partitioned and allocated to each port of the memory controller that can meet the demanded bandwidth. Note that the configuration of virtual clusters changes adaptively with the control cycle.

For example, the zth cluster G_z at port p_z allocating bandwidth $B(p_z)$ is formed by

$$G_z = \{c_i | B_d(c_i) \leq B(p_z);\ c_i \in C,\ z = 1, 2, \ldots, M\}. \quad (5)$$

Such a clustering by the magnitude of memory access is called space multiplexing. To handle the large clustered bandwidth, cores in one cluster can be further assigned with priorities to access memory in a time-multiplexed manner.

Time multiplexing: Time-slot allocation

The time-slot allocation for time multiplexing can be described as follows. Memory-access requests from cores can arrive at different instants and can have a different memory-access number. The core with the earliest request will access the I/O channel first.

The core with early request arrival will be assigned high priority, given as

$$R_i = H, \qquad \text{if } t_a(c_i) < t_a(c_j) \quad (6)$$

where H indicates the high priority flag, and $t_a(c_i)$ indicates the arrival time of the request from core c_i.

However, when multiple requests arrive at the same instant, the core with the highest memory-access number in that control cycle is assigned high priority to access the I/O channel, given as

$$R_i = H, \quad \text{if } M(c_i) > M(c_j) \text{ and } t_a(c_i) = t_a(c_j). \quad (7)$$

Thus, the I/O channel is allocated to a core only if the core is assigned with the highest priority. When all the cores are running with the same workload of benchmark, then multiplexing will be performed in a random manner due to similar benchmark characteristics. However, when some of the cores are idle, the idle time slots will be used by other cores with the aid of assigned memory-access pattern-based priority. It needs to be noted that the allocation of I/O channels is adaptive, achieved by a reconfigurable crossbar switch at the memory controller.

Space-time multiplexing-based demand–supply matching

The space-time multiplexing-based 2.5-D I/O channel management inside the memory controller is given in Algorithm 1.

Algorithm 1 I/O Management by space-time multiplexing

Require: Set of cores C, ports P, bandwidth demands $B_d(c_i)$

1: **for** $i = 1:\ N_c$ **do**
2: $\quad G_z = \{c_i | B(p_{z-1}) < \sum B_d(c_i) \leq B(p_z);\ z = 1, 2, \ldots, M\}$
3: **end for**
4: channel allocation for each cluster
5: **for** $z = 1:\ M$ **do**
6: $\quad$ With all cores inside G_z
7: $\quad$ **for** $t = 1:\ T_c/T_s$ **do**
8: $\quad\quad$ Update priority for each core
9: $\quad\quad$ **if** there is only one request from core at the instant **then**
10: $\quad\quad\quad$ Assign channel to this core
11: $\quad\quad$ **else**
12: $\quad\quad\quad$ Assign channel to the core with the highest priority
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: **end for**

Ensure: Clusters G, allocated channels, and time slots

Initially, memory controller ports P are labeled in the ascending order of the bandwidth that can provided, i.e., $B(p_z) > B(p_{z-1})$. Cores with similar bandwidth demands are grouped into one cluster and connected to one port with required 2.5-D I/O channels (lines 1–4 of Algorithm 1). Once the cores are clustered, the priority-based scheduling will be performed at each time slot. The requests from cores will be processed on a first-come–first-serve basis. At a time slot, if there is only one request, the

corresponding core will be served first (lines 9–10). If there are more than one requests, then the core with more requests is assigned higher priority and served (line 12). Here, "served" indicates that the 2.5-D I/O channel will be occupied by the core in need. This process is repeated for every control cycle T_c. Thus, I/O channel management can be performed in a space-time-multiplexed manner to improve I/O bandwidth utilization and QoS.

Experimental results

Experimental setup

A system-level cycle-accurate simulation is performed on Gem5 simulator [11] for up to 64-core 1 GHz × 86 microprocessors with L2 cache of 256 KB and L1 I and D cache of 32 KB and 64 B cache line. The delay and power models of 2.5-D TSI I/Os are introduced in the simulator. DrSim simulator [12] is employed to implement the proposed space-time multiplexing for 2.5-D I/O management.

For the purpose of I/O management, control cycle T_c and time slot T_s are set as 1 and 0.1 ms, respectively. These values are set based on the switching time of the crossbar switch. The possible overheads of the proposed implementation can be the crossbar switch network in the memory controller and the extra logic needed to perform the scheduling. Area overhead due to the memory controller is 0.187 mm^2 for a 64-core microprocessor under a 32-nm technology node. The area overhead is calculated from a modified memory controller implementation using Xilinx IP. To analyze the efficiency of the proposed memory controller, benchmarks from the general-purpose SPEC 2006, Parsec, and Phoenix are used. The main memory is a DRAM chip, working at clock frequency of 800 MHz with capacity of 4 GB and DDR3-1066 dual channel.

Memory-access data-pattern analysis

Memory-access data patterns can be classified based on the bandwidth demands (LLC and the number of memory requests), as shown in Figure 4a. We classify benchmarks as high, medium, and low memory-access benchmarks. For example, a core running the *lbm* benchmark can be classified

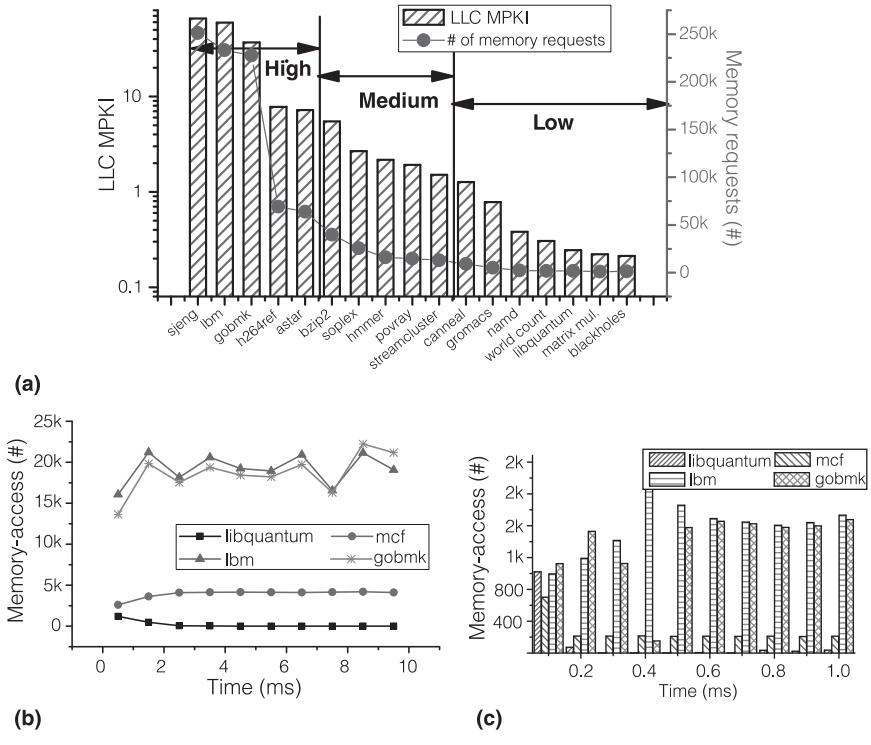
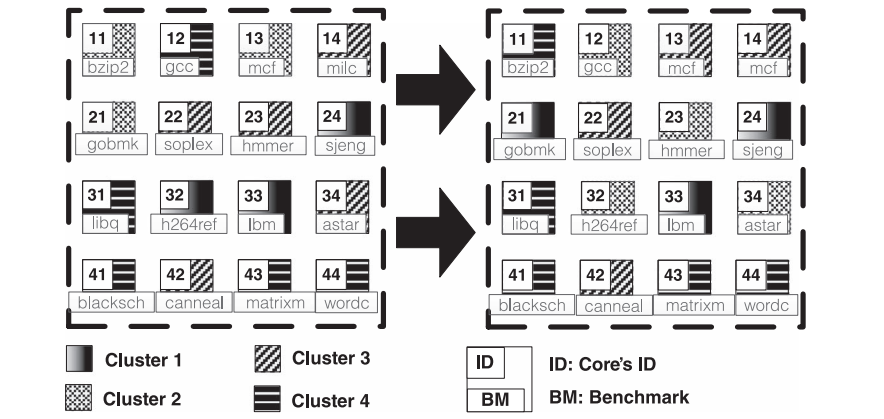
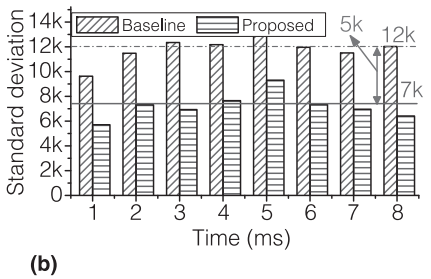
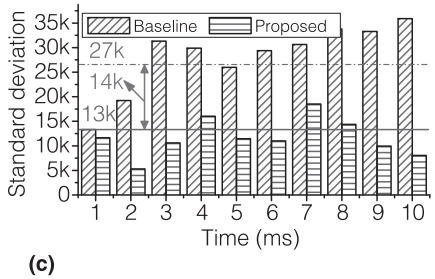
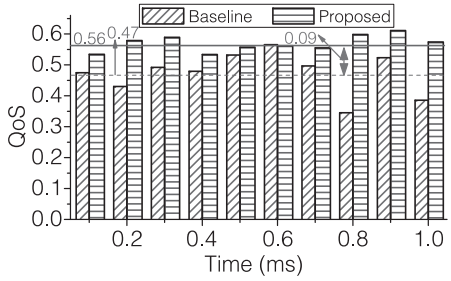


Figure 4. (a) Analysis of benchmarks; SPEC 2006 memory-access data patterns for: (b) execution time of 10 ms; and (c) one control cycle of 1 ms.









For 16-core microprocessor

Method	Requests served	QoS
Baseline	448578	0.473
Time multiplexing	523840	0.528
Proposed	580399	0.589

For 64-core microprocessor

Method	Requests served	QoS
Baseline	2092083	0.461
Time multiplexing	2340843	0.514
Proposed	2622005	0.577

(e)

Figure 5. (a) Adaptive clustering of cores at two consecutive control cycles. (b) Bandwidth balancing for the 16-core by the proposed I/O management. (c) Bandwidth balancing for the 64-core by the proposed I/O management. (d) Communication QoS improvement by the proposed I/O management for the 16-core. (e) Improvement in QoS and the number of requests served.

under high memory access based on its LLC MPKI value, whereas a core running the *libquantum* benchmark will be classified under low memory access due to low LLC MPKI. One needs to observe that LLC and the number of memory-access requests are proportional. In the current work, we consider multiprogram workloads and multithreaded applications and their distribution cores as future work.

For the ease of understanding, consider benchmarks with high and low memory-access intensity and study their memory-access patterns. From Figure 4b, one can observe that *lbm* and *gobmk* have similar bandwidth signatures and high memory-access demands compared to others. Thus, *lbm* and *gobmk* can form a cluster (space multiplexing). Figure 4c presents the memory-access number within one control cycle. Variation in the number of memory-access demands of cores in one cluster (*lbm* and *gobmk*) can be observed, upon which priorities can be assigned to occupy I/O channels in time-multiplexed manner.

Adaptive clustering analysis

Based on the memory-access characteristics, we present the adaptive clustering of cores. A baseline system is considered with static connections, with each port connected to four cores and two I/O channels. For the 16-core case, we assume that the total number of I/O channels is eight and the number of ports is four with each core randomly assigned with a benchmark. For the 16-core case, with the proposed system, ports are allocated with 4, 2, 1, and 1 I/O channels to meet the memory-access demands from high- to low-traffic benchmarks.

Figure 5a illustrates the adaptive clustering result of the 16-core case at two consecutive control cycles. Different filling patterns represent different clusters. Cluster 1 handles high-traffic workloads, cluster 2 is composed of middle-traffic workloads, and so on. Transition of cores from one cluster to the other based on memory-access characteristics can be observed. For example, at the first control cycle, core 12 running the *gcc* benchmark is assigned to low-traffic cluster 4, but it is assigned to middle-traffic cluster 2 in the next control cycle due to time-varying memory-access characteristics.

Bandwidth balancing

Bandwidth balancing across all ports can improve the I/O channel utilization efficiency. We use requests per channel to measure the traffic flow at each control cycle and calculate the standard deviation across all ports. The standard deviation shows how much variation there is from the average value. A low standard deviation indicates more bandwidth balancing. Bandwidth balancing for 16-core and 64-core microprocessors, with randomly distributed SPEC 2006, Parsec, and Phoenix benchmarks to cores is shown in Figure 5b and c. For example, in the 64-core microprocessor, at 5 ms, the baseline system shows the standard deviation of 26 K, while the proposed system has a deviation of 11.5 K with 55.90% improved bandwidth balance. On average, the proposed method can improve the bandwidth balancing by 58.25% under the 16-core case and 58.85% under the 64-core case, compared to the baseline method.

QoS analysis

For a 16-core microprocessor to analyze the QoS, SPEC 2006, Parsec, and Phoenix benchmarks are randomly distributed on cores. Improvement in QoS within one control cycle by the proposed system is presented in Figure 5d. The average QoS for the baseline system, i.e., the static memory controller such as modified implementation of [7] is 0.472, whereas for the proposed algorithm, it achieves a QoS of 0.561. Further considering a 10-ms span as an example, the average QoS achieved by proposed, baseline, and time multiplexing (priority-based scheduling) is 0.589, 0.473, and 0.528, respectively, shown in Figure 5e. This indicates nearly 11.90% higher QoS by the proposed method compared to baseline. Additionally, the number of requests serving the memory controller is also presented. For a 64-core microprocessor, the proposed method achieves a QoS of 0.577 compared to 0.461 and 0.514 achieved by baseline and time multiplexing techniques.

Thermally resilient memory-logic integration by 2.5-D TSI I/Os is addressed in this paper with comparison to 3-D TSV I/Os. In order to further boost 2.5-D I/O communication bandwidth, a data-pattern-aware reconfigurable memory controller is demonstrated by space-time multiplexing, which intends to match huge demand from many cores to access one shared memory. By adaptive clustering cores upon bandwidth, 2.5-D I/O channels are allocated to core clusters by space multiplexing. With further considering the priority, time slots are allocated to cores in one cluster to occupy channels by time multiplexing. The proposed 2.5-D architecture is verified by the system-level simulator with benchmarked workloads. The results show up to 58.85% bandwidth balancing and 11.90% QoS improvement. ■

Acknowledgment

We would like to acknowledge the funding support from MOE Tier-2 (Singapore), A*-STAR PSF Fund (Singapore) and Intel Research Lab (USA). Dongjun Xu and Sai Manoj P. D. contributed equally to this work.

■ References

[1] M. B. Healy et al., "Design and analysis of 3D-MAPS: A many-core 3D processor with stacked memory," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2010, DOI: 10.1109/CICC.2010.5617464.

[2] A. Vassighi and M. Sachdev, "Thermal runaway in integrated circuits," *IEEE Trans. Device Mater. Reliab.*, vol. 6, no. 2, pp. 300–305, Jun. 2006.

[3] S.-S. Wu et al., "A thermal resilient integration of many-core microprocessors and main memory

by 2.5D TSI I/Os," in *Proc. ACM/IEEE Design Autom. Test Eur. Conf. Exhibit.*, 2014, DOI: 10.7873/DATE.2014.190.

[4] M. Zhu, J. Lee, and K. Choi, "An adaptive routing algorithm for 3D mesh NoC with limited vertical bandwidth," in *Proc. IEEE Int. Conf. VLSI System-on-Chip*, 2012, pp. 18–23, DOI: 10.1109/VLSI-SoC.2012.6378999.

[5] J. R. Cubillo et al., "Interconnect design and analysis for through silicon interposers (TSIs)," in *Proc. IEEE Int. 3D Syst. Integr. Conf.*, 2012, DOI: 10.1109/3DIC.2012.6263039.

[6] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," *ACM SIGARCH Comput. Arch. News*, vol. 28, no. 2, pp. 128–138, May 2000.

[7] S. Bayliss and G. A. Constantinides, "Methodology for designing statically scheduled application-specific SDRAM controllers using constrained local search," in *Proc. IEEE Int. Conf. Field-Programm. Technol.*, 2009, pp. 304–307, DOI: 10.1109/FPT.2009.5377664.

[8] M. Paolieri, E. Quiones, F. J. Cazorla, and M. Valero, "An analyzable memory controller for hard real-time CMPs," *IEEE Embedded Syst. Lett.*, vol. 1, no. 4, pp. 86–90, Dec. 2009.

[9] K. Sewell et al., "Swizzle-switch networks for many-core systems," *IEEE J. Emerging Sel. Top. Circuits Syst.*, vol. 2, no. 2, pp. 278–294, Jun. 2012.

[10] S. P. Muralidhara, L. Subramanian, O. Mutlu, M. Kandemir, and T. Moscibroda, "Reducing memory interference in multicore systems via application aware memory channel partitioning," in *Proc. IEEE/ACM Int. Symp. Microarchitect.*, 2011, pp. 374–385, DOI: 10.1145/2155620.2155664.

[11] N. Binkert et al., "The gem5 simulator," *SIGARCH Comput. Arch. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[12] M. K. Jeong, D. H. Yoon, and M. Erez, "DrSim: A platform for flexible DRAM system research." [Online]. Available: http://lph.ece.utexas.edu/public/DrSim

Dongjun Xu started working as a Research Assistant at the Nanyang Technological University, Singapore, as part of his PhD work. His research interests include 3-D ICs, multicore, and low-power designs. Xu has a BS from Xi'an University of Technology, Xi'an, China (2010), where he later continued to pursue a PhD. He is a Student Member of the IEEE.

Sai Manoj P. D. joined Nanyang Technological University, Singapore, in 2012. His research interests include 3-D TSV and 2-D TSI I/O modeling, system-level power management, machine learning, and network-on-chips. Manoj P. D. has an MTech from the International Institute of Information Technology (IIIT), Bangalore, India (2012). He is a Student Member of the IEEE.

Kanwen Wang has been a Research Fellow at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, since April 2012. His primary research interests include cyber–physical power management and 2.5-D/3-D system architectures for exa-scale computing. Wang has a PhD in microelectronics from Fudan University, Shanghai, China (2012).

Hao Yu has been an Assistant Professor at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, since October 2009. His primary research interests are 3-D ICs and RF ICs at nanoterascale. Yu has a PhD from the Electrical Engineering Department, University of California Los Angeles, Los Angeles, CA, USA. He is a Senior Member of the IEEE.

Ningmei Yu is currently a Professor in the Department of Electronic Engineering, Xi'an University of Technology, Xi'an, China. Her research interests include 3-D ICs, multicore, and low-power design. Yu has a PhD from Tohoku University, Sendai, Japan (1999).

Mingbin Yu is working in A*STAR Institute of Microelectronics, Singapore. His current research interests include nanomaterials in optical and electrical application, silicon electronic–photonic devices, and integrated circuit technology. Yu has a PhD from Xi'an Jiaotong University, Xi'an, China (1995).

■ Direct questions and comments about this article to Sai Manoj P. D., School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798.