Filter router: An enhanced router design for efficient stacked shared cache network

Huatao Zhao¹,2,a), Xu Jia², and Takahiro Watanabe¹

Abstract In this paper, many shared cache accesses such as crossed accesses and repeated accesses will be filtered in proposed router network for purpose of access latency reduction. Firstly, the distribution features of all shared cache accesses are analyzed for further optimization on access latency. And then, a meshed router network integrated with enhanced routers is proposed for fast identification of target accesses and further handling them. Hence, the experimental results show that our network design can achieve an average improvement of 26.1 percent on speedup IPC and an average saving of 9.7 percent on energy consumption over base system.

Keywords: stacked integration, shared cache memory, network on a chip, router architecture, TSVs

Classification: Integrated circuits

1. Introduction

Modern processors tend to integrate dozens or even hundreds of cores within single chip for purpose of high speed parallel computing [3, 16, 20]. However, many cores and their private caches are concentrating on the only shared cache, which may lead to serious interconnecting problem. In addition, their physical distances between cores and cache hierarchies are too large to efficiently transmit vast quantities of data [4, 9]. Fortunately, the stacked on chip architectures are recently proposed to solve such interconnection problem via piling each core on adjacent cache banks in the third dimension [10, 21]. Hence, physical distances between cores and cache banks can be largely shortened with the aid of Through Silicon Vias (TSVs) [14, 18].

Although many recent researches discover that stacked architectures are greatly adapted in area saving, network interconnection and layout optimization [8, 21], however, those architectures are limited in their ability to match locality distributions among applications, and to manage highly shared data efficiently as each application contains different behaviors on runtime system latency, performance and energy debit [29]. Moreover, situations are more critical in shared last level cache, because the shared cache should serve too many threads for many data sharing, resulting in serious efficiency and coherence problems [15, 17, 26]. For better management of the shared cache, partitioned cache methods [1, 7, 22] are proposed, which can allocate cache parts into several groups corresponding to each thread.

In this paper, aiming at shared cache access problems, an enhanced router architecture is proposed to form the efficient shared cache network based on a three-layer stacked 3D system architecture. Firstly, shared cache accesses except miss accesses are experimentally classified into six types, and those access types can be fully utilized for access latency reduction according to their particular distribution features. Secondly, proposed router is enhanced with a concurrent path, and in this path, many shared cache accesses can be explored to match with previous hit accesses across the whole router network, and as a result, those filtered accesses can be handled with very low access latency rather than shared cache access latency or even off-chip access latency. Thirdly, proposed router is implemented to generate layout report which depicts little hardware overhead. And experimental results show that our proposed system is comparatively much better on both speedup IPC and energy consumption over base system.

2. Issue description and motivation

2.1 Cases study for expressing access issues

Fig. 1 represents an example diagram of proposed architecture, where each core transports data between first layer and middle layer through a linked router. Meanwhile, all routers together can easily form a meshed network for parallel data interconnection. And all shared cache banks are equally partitioned into several bank groups while each private core unit is allocated with one bank group (links to one router).

**Fig. 1.** An example diagram of proposed stacked 3D on-chip architecture.

¹Graduate School of IPS, WASEDA University, Wakamatsu-ku, Fukuoka 808–0135, Japan
²Electrical Academy, Jiangxi University of Science and Technology, Ganzhou 341–0000, China

a) zhaohuatao@ruri.waseda.jp

DOI: 10.1587/elx.16.20190358
Received May 30, 2019
Accepted June 10, 2019
Publicized July 1, 2019
Copyedited July 25, 2019

Copyright © 2019 The Institute of Electronics, Information and Communication Engineers
Thus, in spite of pure cache miss accesses (miss in entire shared cache), all shared cache accesses can be counted into six types based on the existing location of previous target data: repeated read/write, crossed read/write, shared read/write, where the words (repeated, crossed and shared) stand for hitting only at its own partitioned bank group, hitting only at other bank groups (miss in current group) and hitting at both current group and other groups, respectively. And processing paths of those access types can be described as follows.

(1) Crossed read. For example, DataX is only existing in Bank1 (belongs to Core0), and later, CoreN needs to load DataX. In the partitioned shared cache, the CoreN will get an access miss in its partitioned bank group and such miss may waste many cycles for loading DataX from off-chip memory (or loading in virtue of directory), although the DataX is existing in shared cache.

(2) Crossed write. For example, CoreN will write DataX into its bank group while old DataX is existing in other groups. In the partitioned shared cache, data incoherence may be happened between current data and old data, and then many cycles are consumed to maintain DataX.

(3) Shared read and write. For example, DataX is existing in bank groups of both CoreN and Core0. A shared read can be handled by a normal shared cache access, however, a shared write may cause data incoherence among bank groups.

(4) Repeated read and write. For example, CoreN will access its bank group again on stored DataX (only exists in current group). And both shared read and write can be handled by a normal shared cache access.

Based on the above-mentioned access type descriptions, many potential improvements can be achieved by bridging each bank group with the ability of mutual data transmission. For example, a crossed read can get the target data from other bank groups. Quite sensibly, each router in the first layer acts as the usable component which supports fast data interconnection in the router network and also supports data routing between private core unit and shared cache. Hence, we are motivated to enhance each router with the ability of recording recently handled accesses, data replacement and data transmission.

As a consequence, the target data of a crossed read can be delivered from other bank groups, thereby avoiding an off-chip memory access. The target data of both a crossed write and a shared write can be employed for updating old data of other bank groups with the help of router network, thereby avoiding complex process of incoherence maintenance. Furthermore, the accesses of shared read, repeated read and repeated write can be quickly handled if the target data is stored in the router rather than the processing path of shared cache access.

Moreover, the efficiency of enhanced router is highly related on the distributions of each access type, which will be described in the next subsection.

2.2 Access distributions on shared cache accesses

To analyze access distributions of shared cache, the PARSEC benchmark suit [15] is employed to run on the simulation platform in which simulation setups are same as the ones described in Section 5.

As shown in Fig. 2(a), hit distributions of shared write, shared read, crossed write, crossed read, shared write and read among benchmarks take the average percentage of 5.2 percent, 18.6 percent, 3.6 percent, 9.7 percent, 19.1 percent and 43.8 percent, respectively. Thus, great improvements will be achieved as sizable percentages of those accesses are handled in proposed router network.

Moreover, counted reuse percentages are shown in Fig. 2(b), where on average 72.6 percent accesses will be reused with 256-entry record directories. Thus, it is preferable that each enhanced router can store some recent access results for further reusing in the first layer rather than in the down layer, and hardware tradeoff on entry number can be set as 256-entry for each router.

To sum up, the above access distributions indicate the potential improvement if access paths of some types are optimized. And we are motivated to bridge those filtered shared cache accesses with much faster processing paths with the aid of enhanced routers. Moreover, those enhanced routers can form a router network to fast trace target accesses during runtime and further filtered accesses can be rapidly handled in a concurrent path, thereby achieving many improvements on performance.
3. Proposed stacked 3D on-chip network

3.1 Base network architecture

To compare with our proposed network design, we firstly describe the base system based on router model described in [19]. As shown in Fig. 3, a private core unit including a core and a private cache is linked to a router in the first layer, and the router stretches out a TSV to link with a partitioned bank group including cache controllers and directory controllers in the middle layer. As to data sharing, we employ the directory-based mechanism to maintain shared cache coherence across bank groups [30]. In addition, to interconnect data among groups, the dynamic non-uniform cache architecture (D-NUCA) is selected to form the shared cache in purpose of splendid cache line migration [27]. Moreover, each bank group has a router in the middle layer, which is employed to route off-chip memory accesses in the bottom layer.

3.2 Enhanced router architecture

Fig. 4 shows the internal architecture and routing pipeline of our proposed router. Along with conventional routing path, the modified path of proposed router is independent to work concurrently with conventional router flow. The modified path has three key components which are depicted as follows:

1. Target Explorer (TE). As shown in Fig. 4(c), the explorer is applied to fast search each target access and to record each recent access result. Firstly, we employ a fast lookup buffer to receive a target access and then actuate a look-up at the Record Entry Table (maintained under LRU replacement policy). Meanwhile, the bypass unit is employed to spread the current target access forward to all of other routers by broadcasting across the routing network [13]. If there is a hit on any record entry tables of all routers, linked data block can rapidly return the target data to private core unit or can be rapidly handled within router network. Otherwise, the requested data block is obtained by critical path.

2. Replacement Coherence logic (RC). To maintain data coherence across the router network, a directory based mechanism with full bit vector format is employed to track data status (exclusive, shared or modified) in each record entry and linked data block. Once there is a data shared (including shared and crossed accesses) in record entries of different routers, this data should be sent to the RC logic for coherence check. As shown in Fig. 4(d), the coherence logic does three operations that generate coherence order to update the target data to the linked data block, to update the target data to the partitioned bank group, or to receive or forward the target data to other routers.

3. Data deliverer (DD). As shown in Fig. 4(e), the deliverer mainly consists of a mux circuit, output drivers and a crossbar switch, and it is designed to perform the following processes concurrently: delivering some target data to the private core unit, unifying some modified writes to record entry table in purpose of data coherence, bypassing some data across to other routers [2], or writing retired data back to the partitioned bank group.

3.3 Routing pass and network topology

In this section, we tend to describe how each target access can be eased in proposed router network. As shown in Fig. 4(b), six access types can be depicted as follows:

1. Repeated read: find previous read in TE stage, then deliver the data to private core unit in DD stage.

2. Repeated write: find previous write in TE stage, then generate write updating order in RC stage, and finally update the linked data block in the DD stage.

3. Shared read: find previous read in TE stage, then deliver the data to private core unit in DD stage.

4. Shared write: find previous write in TE stage, then generate write updating order in RC stage, then update the linked data block and deliver data to output port in the DD stage, then bypass the data to other routers directly in virtue of directory information, then generate write updating orders in other routers, and finally update the linked data blocks in the DD stage.

5. Crossed read: find previous read in other routers, then deliver the data to the output port, then bypass the data to the target router, and finally deliver the data to both private core unit and record entry table in DD stage.

6. Crossed write: find previous write in other routers, then write the data to its linked data block and deliver data to output port in the DD stage, then bypass the data to other routers, then generate write updating orders in other
routers, and finally update the linked data blocks of other routers.

To pass through the router network, we employ the meshed XY routing method, where the x-direction lines are endowed with high priority and y-direction lines are given low priority [12].

As we only intend to filter some accesses in the router and the record entries of each router are very few, each router can establish a full bit vector based directory for pointing data sharing among routers. Note that the replacement coherence logics in router network are independently functioned over the directory-based coherence mechanism of shared cache, and thereby each write in the router network will finally be delivered to the partitioned bank groups, and then the coherence mechanism of shared cache starts to maintain data coherence in shared cache.

4. Layout verification

4.1 Layout setups
To verify the hardware overhead of proposed router, we implement the router through verilog modeling, timing analysis (ModelSim10.4), synthesis and layout (Synopsys IC Compiler). We enhance the proposed router based on the conventional router models [11, 19].

4.2 Router hardware analysis on area, energy and latency
Fig. 5 shows the router layout figure implemented in a 1.3 × 1.3 mm² board. As shown in Table I, our five-level logic router has 165594 gates counted, compared to a 16 MB shared cache, the integration scale of one router is about 0.681 percent of a 16 MB shared cache (generated in CACTI v6.5) [24]. And design area counted in 16 routers is about 0.681 percent of a 16 MB shared cache, the integration scale of one router is 45 nm technology library, 30 cycle TLB miss latency.

As shown in Fig. 4(b), access latency of conventional shared cache access is about 20 clock cycles (10 ns) which can be regarded as average access latency of critical path.

Once there is a hit on record entry table, current shared cache access can be handled rapidly with the router network in the first layer.

5. Experimental evaluation

5.1 Platform setups
Platform model: We employ the detail-emulated, cycle-precise gem5 simulator [23] to constitute a multi-core processor model, and the processor parameters are listed in Table II. To pass across three layers, we use 16 TSVs between two layers while the TSV model is based on the previous work [5]. We achieve both performance and energy consumption with the aid of PIN tool [25].

Router model: Based on layout reports, we interlink the router details including latency and power results into the platform to establish the proposed router network.

Benchmarks: we employ PARSEC benchmark suite as the multi-thread workloads [15]. We run each benchmark to skip initial one billion dynamic instructions, and next we count five billion dynamic instructions uniformly to calculate the speedup IPC and on-chip energy.

| Name                | Quantity          | Values from IC Compiler |
|---------------------|-------------------|-------------------------|
| Layout Setsups      | Clock frequency   | 533 MHz                 |
|                     | Supply voltage    | 0.9 V                   |
|                     | Technology library| 45 nm                   |
| Statistic Reports   | Levels of Logic   | 5                       |
|                     | Number of ports   | 4852                    |
|                     | Cell Count        | 24715                   |
|                     | Gate Count        | 165594                  |
| Area Reports        | Combinational Area| 0.532 mm²               |
|                     | Cell Area         | 0.899 mm²               |
|                     | Net Area          | 0.625 mm²               |
|                     | Design Area       | 1.504 mm²               |
| Power Reports       | Cell Internal Power| 31.5495 mW             |
|                     | Total Dynamic Power| 56.9384 mW             |
|                     | Cell Leakage Power| 11.4590 mW              |
| Clock Reports       | Clock Sinks       | 4263                    |
|                     | Longest Path      | 0.278 ns                |
|                     | Total DRC         | 17                      |

Table II. Test system configurations.

| Component     | Configurations                               |
|---------------|----------------------------------------------|
| Processor     | 2.0 GHz, 16 cores, single thread per core, 1.1V supply voltage, 128 IW entries, 45 nm technology library, 30 cycle TLB miss latency. |
| Unit          | L1/L1D Cache 16KB for a core (private) including 16KB L1I cache and L1D cache, 4-way, 64 B line size, 2 cycle latency. |
|               | L2 Cache 16MB total size D-NUCA, 128 banks, 1 MB partitioned for a core (shared), 8-way, 128 B line size, 20 cycle latency. |
| Memory        | Main 4GB Double Data Tape (DDR4 2133 MHz, 1.2V), 8KB page size, 120 cycle latency. |
| Router Network| Mesh.XY Topology, 5 ports per router, 3 VC's per port, 15 flits per port. |
Comparisons: Our proposed router network is compared with three methods: base system, base system with 64 KB L1 cache and proactive resource allocation (PRA) based method [6]. Those methods are simulated with same setups as our proposed system, except that they are not constituted with the concurrent routing path.

Evaluating metrics: Our evaluation metrics are set as the IPC and energy consumption. All group of experiments are uniformed with same amount of workloads. As shown in Eq. (1), we define the speedup IPC as the sum of the IPC in all cores, and energy consumption is counted within on-chip components.

\[
\text{Speedup IPC} = \frac{1}{N} \sum_{i=1}^{N} \text{IPC}_{\text{Core}}^i
\]

\[
\text{On-chip Energy} = \sum \text{Energy}_{\text{on-chip component}}
\]

5.2 Experimental results and discussions
Performance improvement: Fig. 6 shows the normalized speedup IPC of the proposed system with record entry table size ranging from 32 to 2 K, where the IPC values of base system are set as 1. Our proposed system can outperform the base system an average of 4.2 percent (the lowest, 32 entries), 26.1 percent (the highest, 256 entries) and 24.3 percent (2 K entries) speedup IPC improvement, respectively. As to each highest improvement value among different benchmarks, it ranges from 17.4 percent (swaptions, 512 entries) to 39.5 percent (x264, 256 entries).

To analyze the speedup IPC improvements, two key factors should be considered at first: 1) The amount of filtered shared cache accesses, 2) The latency of target exploration increases in proportion with the record entry scale. The former factor is highly related to the entries scale. As shown in Fig. 2(b), the amount of filtered accesses is rising first sharply and then steadily. Since the working sets of workloads are limited in size, adding more entries at first can bring in more filtered accesses, and further the speedup IPC will be increased. Later, an addition of extra 1 K record entries only results in a few extra filtered accesses. But even worse, large record entries will cause a long latency on exploring target accesses, and further cut down some filtered path improvement. Hence, there has been a tradeoff between those two factors and also the suitable entry scale can be explored by weighing IPC improvement.

Energy saving: Fig. 7 shows the on-chip energy values of the proposed system which are normalized to the values of base system. Note that those values are calculated during the execution period of five billion dynamic instructions, where each test is unified with the same amount of workloads. In addition, our proposed system will complete the instruction execution much faster than base system, and then it can save some energy since its routers are enhanced with extra hardware. Experimentally, our proposed system consumes extra 0.6 percent on-chip energy over base system in case of allocating 32 record entries, because the enhanced routers have larger hardware scale. At the same time, proposed system can save an average of 9.7 percent in case of allocating the 256 record entries, because our proposed system works much faster than base system.

5.3 Comparisons with related work
Fig. 8 also shows the speedup IPC and on-chip energy values of base system, base system with 64 KB L1 cache, PRA based system and the proposed system (256 record entries). The PRA based system can achieve the speedup IPC improvement ranging from 7.4 percent (facesim) to 16.8 percent (vips) over base system, meanwhile, our proposed system on speedup IPC can outperform base system, base system with 64 KB L1 cache and PRA based system by average 26.1, 26.8 and 14.3 percent, respectively. Moreover, on-chip energy values of the proposed system can be saved by average 9.7, 10.5 and –1.2 percent over three methods, respectively.
6. Conclusion

In this work, we propose a stacked 3D cache scheme with a router network to handle shared cache access. The router network serves to filter six kinds of shared cache accesses, and further those accesses can be handled with very low latency. Experimental results show that our proposed stacked on-chip solution can achieve an average of 26.1 percent speedup IPC improvement and save an average of 9.7 percent on-chip energy consumption compared to base system.

Acknowledgments

We would like thanks Mr. Zhang Xiaoyang (Synopsys Shanghai) who had fully supported on router layout. This work is partly supported by GJJ180486.

References

[1] K. Huang, et al.: “Curve fitting based shared cache partitioning scheme for energy saving,” IEICE Electron. Express 15 (2018) 20180886 (DOI: 10.1587/elex.15.20180886).
[2] P. Guo, et al.: “A bypass-based low latency network-on-chip router,” IEICE Electron. Express 16 (2019) 20181147 (DOI: 10.1587/elex.16.20181147).
[3] N. Kim, et al.: “Benzene: An energy-efficient distributed hybrid cache architecture for manycore systems,” ACM Trans. Archit. Code Optim. 15 (2018) 10 (DOI: 10.1145/3177963).
[4] Z. C. Wang, et al.: “VP-router: On balancing the traffic load in on-chip networks,” IEICE Electron. Express 15 (2018) 20180883 (DOI: 10.1587/elex.15.20180883).
[5] T. T. Lu, et al.: “TSV-based 3-D ICs: Design methods and tools,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 36 (2017) 1593 (DOI: 10.1109/TCAD.2017.2666604).
[6] P. Lotfi-Kamran, et al.: “Near-ideal networks-on-chip for servers,” HPCA (2017) 277 (DOI: 10.1109/HPCA.2017.16).
[7] G. Aupy, et al.: “Co-scheduling HPC workloads on cache partitioned CMP platforms,” CLUSTER (2018) 335 (DOI: 10.1109/CLUSTER.2018.00052).
[8] P. Safayenikoo, et al.: “An energy efficient non-uniform last level cache architecture in 3D chip-multiprocessors,” ISQED (2017) 373 (DOI: 10.1109/ISQED.2017.7918344).
[9] V. Srinivasan, et al.: “H3 (Heterogeneity in 3D): A logic-on-logic 3D-stacked heterogeneous multi-core processor,” ICCD (2017) 145 (DOI: 10.1109/ICCD.2017.30).
[10] A. K. Mishra, et al.: “Architecting on-chip interconnects for stacked 3D STT-RAM caches in CMPs,” ISCA (2011) 69 (DOI: 10.1145/204723.2000074).
[11] G. H. Zheng, et al.: “Design and implementation of a NoC router supporting multicast,” IEICE Electron. Express 11 (2014) 20130655 (DOI: 10.1587/elex.10.20130655).
[12] Y. S. Jeong and S. E. Lee: “Deadlock-free XY-YY router for on-chip interconnection network,” IEICE Electron. Express 10 (2013) 20130699 (DOI: 10.1587/elex.10.20130699).
[13] X. Han, et al.: “A deadlock-free subnetwork mechanism for high performance broadcasting in NoC,” IEICE Electron. Express 12 (2015) 20150688 (DOI: 10.1587/elex.12.20150688).
[14] Y. Yang, et al.: “New coaxial through silicon via applied for three dimensional integrated circuits,” IEICE Electron. Express 13 (2016) 20160192 (DOI: 10.1587/elex.13.20160192).
[15] C. Bienia: “Benchmarking modern multiprocessors,” Ph.D. Thesis, Princeton University, Princeton (2011).
[16] K. Aingaran, et al.: “M7: Oracle’s next generation Sparc processor,” IEEE Micro 35 (2015) 36 (DOI: 10.1109/MM.2015.35).
[17] M. R. Marty: “Cache coherence techniques for multicore processors,” Ph.D. Thesis, University of Wisconsin-Madison, Madison (2008).