IPSJ Transactions on Advanced Computing Systems Vol.5 No.1 80–87 (Jan. 2012)

Regular Paper

Adaptive Data Compression on 3D Network-on-Chips

Yuan He[1,a)] Hiroki Matsutani[2] Hiroshi Sasaki[2] Hiroshi Nakamura[2]

Received: May 11, 2011, Accepted: September 11, 2011

Abstract: The three-dimensional Network-on-Chip (3D NoC) is an emerging research topic exploring the network architecture of 3D ICs that stack several wafers or dies. As such topics being extensively studied, it is found negative impacts of 3D NoC's vertical interconnects are raising concerns considering their footprint sizes and routability degradation. In our evaluation, we found such vertical bandwidth limitation can dramatically degrade system performance by up to 2.3×. Since such limitations come from physical design constraints, to mitigate performance degradation, we have no other choice but to reduce the amount of communication data on-chip, especially for those data moving vertically. In this paper, therefore, we carry out a study of data compression on 3D NoC architectures with a comprehensive set of scientific workloads. Firstly, we propose an adaptive data compression scheme for 3D NoCs, taking account of the vertical bandwidth limitation and data compressibility. Secondly, we evaluate our proposal on a 3D NoC platform and we observe that the compressibility based adaptive compression is very useful against incompressible data while the location-based adaptive compression is more effective with more layers for the 3D NoC. Thirdly, we find that in a bandwidth limited situation like a CMP with 3D NoCs having multiple connected layers, adaptive data compression with location-based control or with both compressibility and location based control is very promising if the number of layers grows.

Keywords: 3D Network-on-Chip, data compression, Chip-Multi Processor

1. Introduction

As semiconductor technology progresses, the number of processing cores integrated on a single chip has continually increased. As a proof, commercial/prototype chips that have 64 or more cores have already been produced [18], [19]. Meantime, to meet the increasing demand of on-chip bandwidth, Network-on-Chips (NoCs) [20] have been widely adopted as a replacement of traditional bus-based interconnects for this many-core paradigm.

Recently, the concept of NoCs is being extended to ICs that have three-dimensional structures, namely the 3D NoC [21], in order to mitigate the wire delay and wire energy which are increasingly posing severe problems to modern VLSI design. Traditionally, the wire delay can be mitigated by inserting inverting buffers (i.e., repeaters) on long wires, but the buffers themselves add gate delay and consume energy; thus repeater insertion is not a fundamental solution to the problem. With 3D ICs, a number of wafers or dies are stacked very closely (e.g., 5 μm to 50 μm); thus a 3D structure significantly reduces wire length, wire delay, and wire energy compared to 2D counterparts.

For these reasons, 3D NoC is an emerging research topic, and its network topology [22], router architecture [23], [24], and routing algorithms [25] have already been extensively studied.

However, many studies on 3D IC architectures have underestimated the negative impact of vertical interconnects, as reported in Ref. [5]. Unfortunately, these vertical interconnects, such as through-silicon vias (TSVs) and microbumps, also consume a certain amount of area. In addition, they affect the routability of wires negatively, because some vertical interconnects interfere with metal layers. Thus, although 3D IC technologies are believed sound beyond Moore's Law, their vertical bandwidth is still a major concern. In practice, we find that such vertical bandwidth limitation can severely degrade the system performance by up to 2.3× (see Section 2).

Since vertical bandwidth limitations come from the physical design constraints mentioned above, to mitigate the performance degradation, we have no other choice but to reduce the amount of communication data, especially for those data moving vertically. In this paper, therefore, we carry out a study of data compression on 3D NoC architectures with a comprehensive set of scientific workloads.

The contributions of this paper are the following. Firstly, to the best of our knowledge, this is the first work to characterize and evaluate the effect of data compression on 3D NoCs. Secondly, we are the first to introduce and explore adaptive control of data compression on 3D NoCs. Thirdly, with our evaluation results, we show that the three adaptive compression policies are very promising compared to static compression when applied on 3D NoCs.

The remainder of this paper is organized as follows. Section 2 briefly surveys 3D IC technologies and introduces the 3D NoC model we focus on. Section 3 discusses the compression technique to be used and our adaptive compression scheme to be investigated. The experimental platform, including the simulation model and workloads, is described in Section 4, while Section 5 is devoted to evaluation results and insights into the effects of data

[1] Graduate School of Engineering, The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[2] Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[a)] he@hal.ipc.i.u-tokyo.ac.jp

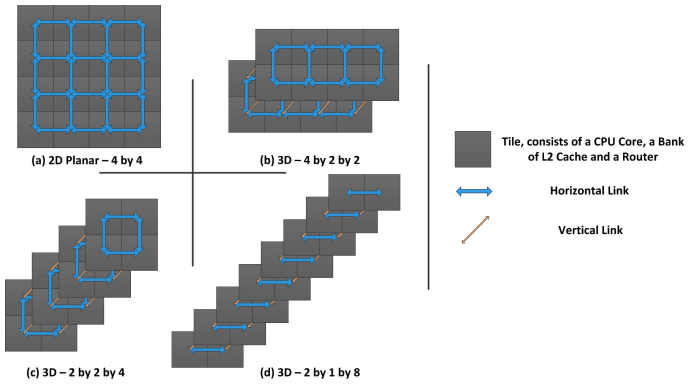

Fig. 1 2D and 3D NoC topologies.

compression on 3D NoCs for CMPs. Section 6 overviews related work, and finally, this paper is concluded in Section 7.

2. 3D NoC Design and Its Limitations

3D ICs bring us many benefits like increased system integration, reduced wire length and increased data locality, but how different wafers or dies are stacked vertically remains an open question for the research community and the industry. Various interconnection technologies of 3D ICs have been developed for the purpose of vertical stacking, such as wire-bonding, micro-bump [1], [2] and through-silicon via (TSV) [3], [4].

- `Wire-bonding` is a die-to-die interconnection formed with bonding wires. It has a footprint recorded from 35 to 100 um [4]. It is the most common approach and has been highly utilized by System-in-Package designs. The limitation is the number of wires and their density as only edges of a chip is used for the purpose of bonding. Obviously, the bonding wire length can be the cause of a considerable communication delay.
- `Micro-bump` forms a die-to-die interconnection through solder balls. It has a footprint known to be from 10 to 100 um [4]. This approach is generally limited to stack only two dies with face-to-face connections but it can also be used to form connections of more than two dies with face-to-back design although this is believed inefficient because of factors like heat.
- `Through-silicon via (TSV)` is a wafer-level interconnection making use of via-holes formed through multiple wafers. The footprint of TSV is 5 to 50 um thus it has the potential of offering a better interconnection density than wire-bonding and micro-bump. However, it suffers from high manufacturing cost due to the fact that an extra process to form these interconnects [4]. Another constraint of TSV comes from routing, as TSV interconnects interfere with gates and wires. So considering yield and cost, the number of TSV interconnects has major impact in design and it should be considered carefully ahead of manufacturing [5].

As briefly explained above, all three interconnection technologies of 3D ICs have a limitation of going vertical, that is, the die-to-die or wafer-to-wafer interconnection can become a bandwidth bottleneck. With larger numbers of such interconnects, we are facing the difficulty of design complexity and cost of manufacturing. To depict this vertical bandwidth limitation, we employ

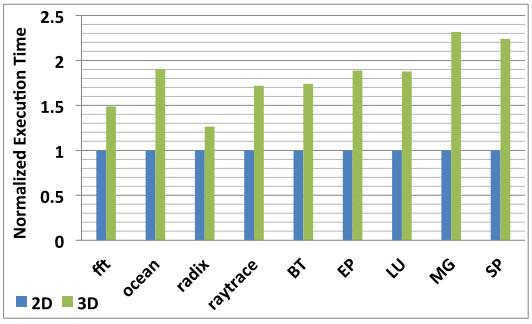

Fig. 2 System performance degradations under link limitations for 3D NoC.

a 3D NoC model with heterogeneous link widths, which is, for vertical links that are used to move data between dies or wafers, we model them as having smaller bit widths compared to horizontal links. In our study, we also try to capture the effects from different numbers of layers (dies/wafers). We have our 3D NoCs modeling 2, 4 and 8 layers. An example of the baseline 2D NoC and three 3D NoC configurations are illustrated in **Fig. 1**. A grey square represents a tile of the modeled NoCs while the blue and orange arrows denote horizontal and vertical links, respectively.

Moreover, **Fig. 2** presents an example of how this link limitation can affect the system performance. Please note the detailed evaluation conditions and environment will be shown in Section 4. Both 2D and 3D NoCs are configured in the same way except their link widths. For this particular evaluation, we tested a 2D NoC having 128-bit links and an 8-layer 3D NoC. For the 3D NoC, its horizontal links are set to 128-bit while its vertical links are 16-bit wide. Both configurations assume a total of 16 cores. In this evaluation, the execution time of the same workload is being increased by up to 2.3×. As shown in Fig. 2, these numbers are far larger than the the 2D NoC with 128-bit links. Thus, vertical link bandwidth limitation is a major bottleneck for any system moving to 3D design.

In our network model, as shown in **Fig. 3** (b), basic building blocks (tile) of our 3D NoCs are connected with each other by routers and links. For comparison purpose, we also include the tile of a 2D design in Fig. 3 (a), whose router is at most having six ports and two of them are used to connect to a processor core and an L2 cache bank. For 3D NoCs, two more ports may be added to the router and through two additional links, different dies/wafers are connected. The network routing scheme is also re-defined since X-Y routing for 2D is not sufficient for the 3D

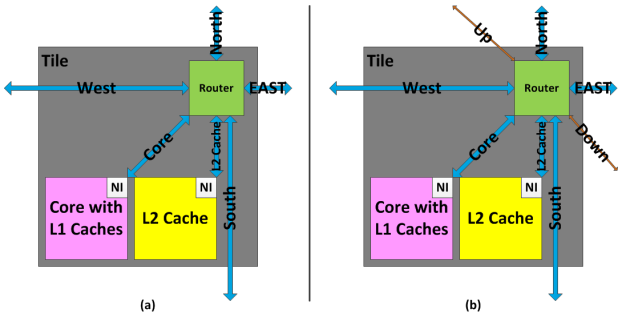

Fig. 3 Tiles of 2D and 3D NoCs.

design. As shown in the last paragraph, because of the layer-to-layer bandwidth limitation, our 3D NoC models narrower vertical links. More details on configurations of the 3D NoC model are covered in Section 4 where we discuss the simulation model.

3. Data Compression on NoCs

Data compression is a popular architectural technique and it has been applied in many fields to conserve on-chip/off-chip bandwidth, to enlarge cache/memory capacity or to reduce communication latency. In our work, we use data compression to conserve bandwidth and to reduce latency for 3D NoCs. In this section, we discuss the compression technique in details. We are going to introduce the compression algorithm, frequent pattern compression, at first. After which, we will focus on implementation issues of this compression algorithm. And finally, our proposal of adaptive control on compression will be presented.

3.1 Compression Algorithm and Implementation

There are several state-of-the-art data compression algorithms which have been applied on NoCs, including frequent pattern compression (FPC) [10] and frequent value compression (FVC) [11], [12]. In this paper, we select FPC because of its simplicity and effectiveness. FPC is a significance-based compression scheme having small compression/de-compression overheads; unlike FVC, it has no synchronization overhead. FPC compresses frequent patterns appeared in data packets. In our case, there are seven such patterns with which we seek to compress each 32-bit of data and a full description of these patterns are presented in **Fig. 4**. Of all these patterns, the selection was made upon their frequencies. In Ref. [10], it is found that zero words, words with 8-bit data and words with 16-bit data are the most frequent patterns for workloads from SPLASH-2 [15] and NPB 3 [16]. Therefore, we selected these patterns as shown in Fig. 4. For all seven data patterns, we assign a 3-bit index to each of them. Along with another index for uncompressed data words, there are in total eight indexes which are the compression overhead. For example, a data word of 32 zeros will be replaced with an index of 000 after compression, while an 8-bit sign-extended data word will be replaced with an index of 001 plus the 8-bit data. Please note that although indexes are fixed to 3-bit, the actual data appended to the index may be different in size. For the last index which is "111," the data is uncompressed which results in a negative effect after the combination of index and data. FPC has advantages of high compression ratio and parallel compression. For 128-bit data, we can always split it into 4 parts and compress all parts with four compression circuits at one time. But since FPC employs variable length compression, the de-compression may have to be done in a serial manner.

Pattern 1: Full Zero Runs
00000000000000000000000000000000 => 000

Pattern 2: 8-bit Sign-extended
00000000000000000000000???????? => 001 + ????????
11111111111111111111111???????? => 001 + ????????

Pattern 3: 16-bit Sign-extended
0000000000000000???????????????? => 010 + ????????????????
1111111111111111???????????????? => 010 + ????????????????

Pattern 4: 8-bit Data Padded with Zeros
????????00000000000000000000000 => 011 + ????????

Pattern 5: 16-bit Data Padded with Zeros
????????????????0000000000000000 => 100 + ????????????????

Pattern 6: Two Half-words, Each as a Sign-extended Byte
00000000????????00000000???????? => 101 + ????????????????
00000000????????11111111???????? => 101 + ????????????????
11111111????????00000000???????? => 101 + ????????????????
11111111????????11111111???????? => 101 + ????????????????

Pattern 7: Word Consisting of Repeated Bytes
[Byte X][Byte X][Byte X][Byte X] => 110 + [Byte X]

Non-compressible Word
[Word X] => 111 + [Word X]

Fig. 4 Patterns of the frequent pattern compression.

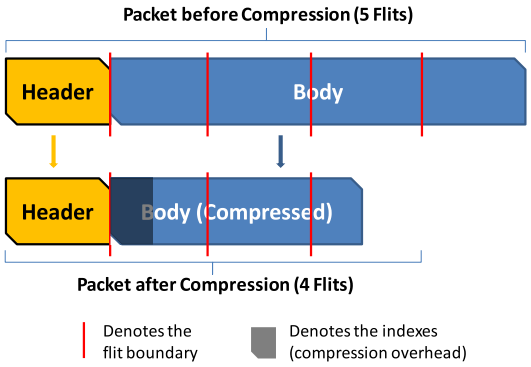

Fig. 5 An example of the frequent pattern compression.

Regarding the implementation, similar to Refs. [10], [11], [12], data compression/de-compression circuits in our work are assumed to be implemented in network interfaces (NI) of our 3D NoCs. At NIs, any injecting data traffic will be compressed and receiving data traffic will be de-compressed; but it is important to note that the enhanced NIs will also have area, latency and energy overheads. The compression and de-compression processes are carried out for data packets only. In our evaluation, any data packet has a 512-bit body which is the size of a cache line. When compression is applied, the 512-bit data is broken into 32-bit pieces, which are then encoded with the eight patterns shown in Fig. 4. If the flit size is 128-bit and the compression ratio is between 25% and 50% (like the one shown in **Fig. 5**), the original packet is composed of a header flit and 4 body flits while the compressed packet carries a header flit and 3 body flits. This results in a 5-flit to 4-flit packet size reduction.

As we mentioned earlier, the compression process of FPC can be done in parallel for several data words at a time. As stated in Ref. [6], this compression process is only taking one cycle per

data word, thus with multiple parallel encoders, the timing overhead of compression is one cycle per packet. For de-compression, since FPC is a variable length compression scheme, it is unable to carry out the de-compression in parallel. But as proposed in Ref. [10], it is able to overlap the network latency with part of this de-compression latency. In details, the receiving and de-compression pipeline is designed to work with only a fraction of a packet received. After the first body flit containing indexes of all compressed words (the compression overhead) is received, there is a pre-computation process in order to obtain the length of compressed data before its arrival. Hence, the de-compression does not need to rely on receiving the entire compressed packet. By applying this improvement, the de-compression timing overhead can be kept within two cycles per packet. Thus, in our evaluation, we assume one cycle of compression delay and two cycles of de-compression delay for any data packet. However, for incompressible packets, their sizes, in terms of number of flits, will be the same or even increased after the compression. This opens up another opportunity for adaptive control in order to avoid negative effects, such as increased packet latency due to having more flits or effortless de-compression.

In Ref. [10], it is recorded that with 45 nm process, the area overhead and dynamic power consumption of compressor/de-compressor circuits are 0.183 mm^2 and 0.273 W, respectively. In our paper, since both the packet size and the compression/de-compression algorithm and process are the same as Ref. [10], we expect a similar area overhead. Power issue is not discussed in our paper and is left for future work.

3.2 Proposed Adaptive Compression for 3D NoCs

In Sections 2 and 3.1, we have discussed our 3D NoC model and its vertical bandwidth limitation. To help mitigating the vertical bandwidth limitation and making better use of FPC, we present an adaptive compression scheme for 3D NoCs. Based on FPC, our adaptive compression scheme utilizes compressibility and location based mechanisms to control the compression process while static FPC employs a constant-on rule that every data packet gets compressed. For any data packet waiting to be injected to the network, we have set up two policies to determine whether the compressor should be invoked or not. There is also a third policy which aggregates these two proposed policies.

- `Compressibility` based control requires the compression process, which incurs overhead of compression. The reason for proposing this policy is that negative compressibility and effortless de-compression should always be avoided. After the actual compression process, we can identify the size of the compressed packet. If it is known that the compressed packet cannot derive any flit reduction from the original packet, then the network interface disregards the compressed packet and instead it splits and injects the original packet. With this policy, for packets whose compressibility is not good enough for any flit reduction, we can save the timing overhead of sending more flits or carrying out an effortless de-compression when compared to static compression. However, if the data is incompressible, we lose one cycle per packet when compared to no compression. When this is the only adaptive control implemented, the compressibility is always checked in spite of the packet direction. **Figure 6** gives three examples of this adaptive control and only the third case has the compression incurred since that packet has less number of flits after compression.

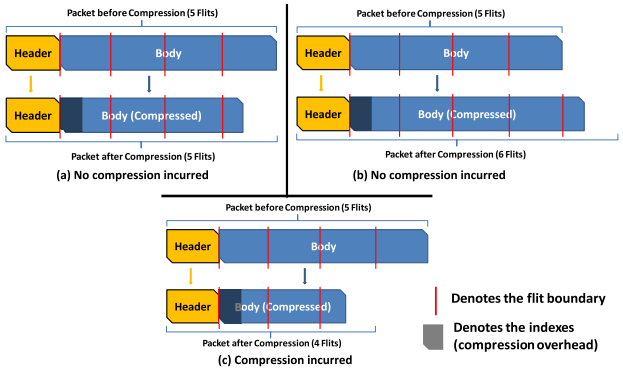

Fig. 6 Compressibility-based adaptive control.

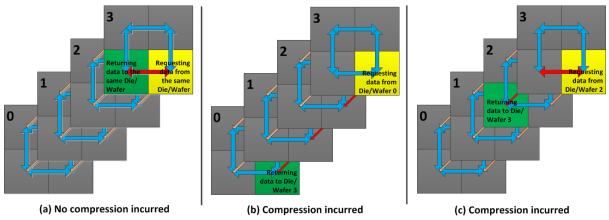

Fig. 7 Location-based adaptive control.

- `Location` based control is simple. It does not require the compression process. As shown in **Fig. 7**, this method detects packets going across layers, such as Fig. 7 (b) and Fig. 7 (c), and compresses them. Layer crossing packets can be easily detected by checking several bits of the packet header indicating the destination node. There are two reasons for proposing this policy. Firstly, we believe 3D NoC will grow with increasing number of layers, which means more traffic will be layer-crossing. Secondly, if most of the compressible packets are crossing layers, compressing these traffic is more promising since they also suffer from the vertical bandwidth limitation as described in Section 2.
- `Compressibility and Location` based control is the logical conjunction of the above two policies. A layer-crossing packet will be examined for compressibility to determine if its compressed form is going to be injected to the network. Please note that packets travelling within the same layer will neither be checked for compressibility nor be compressed. Like the second policy, this policy also targets at the vertical bandwidth limitation. However, it removes any negative compressibility or effortless de-compression for these layer-crossing packets and it also removes the timing overhead of the compressibility check for packets traveling in the same layer. It has one cycle of timing overhead if a layer-crossing packet is incompressible when compared to no compression. Three examples are shown in **Fig. 8**, while only the third one has compression incurred since both conditions are satisfied.

To successfully implement this adaptive control on FPC, it is necessary to have a bit in the header indicating the compression status for all data packets. When compressed, this bit in the

Table 1 Simulation configurations.

Component	Parameter
Processors	16
L1 Cache	Each core has a total of 64 KB of private L1 cache (split I and D), which is 4-way set-associative and has 64 bytes per line and 1 cycle of access latency.
L2 Cache	Shared L2 cache divided into 16 banks. Each bank is 256 KB, 16-way set-associative and has 6 cycles of access latency.
Memory	4 GB of DRAM with 160 cycles of access latency.
Topology	16 nodes organized in three 3D Mesh topologies, 4 by 2 by 2 layers, 2 by 2 by 4 layers and 2 by 1 by 8 layers.
Network Interface	2-stage pipeline for splitting packets into flits and flit injection; and 2-stage pipeline for flit reception and combining flits into a packet. The compression/de-compression circuits are implemented here.
Router	3-stage pipeline with X-Y-Z routing, wormhole switching and 3 virtual channels.
Link	Uneven link width is implemented; the planar link width is 128-bit and the vertical link width is 16-bit.
Compression Overhead	For all compression methods, compression takes 1 cycle while de-compression takes 2 cycles. For compressibility based adaptive policy, the compressibility check is 1 cycle. For location-based adaptive policy, the destination node detection does not cost any additional cycle. Similarly for compressibility and location based adaptive policy, the compressibility check takes 1 cycle but it is only for packets which travel across layers and this destination node detection does not take any additional cycle.

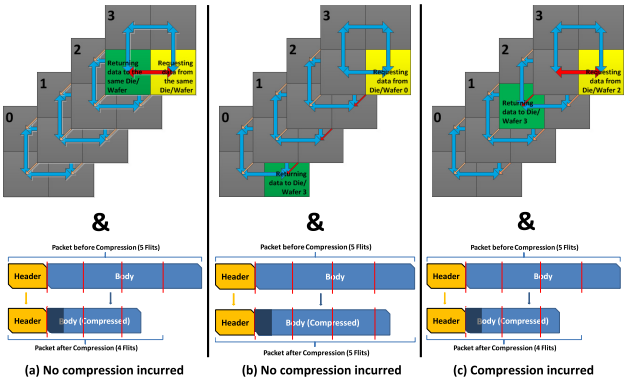

Fig. 8 Compressibility- and location-based adaptive control.

packet header will be set to "1," or this bit is set to "0" when the packet is not compressed.

4. Experimental Platform

In this section, we are going to explain the experimental platform in details. Firstly, we will quantify the parameters of our simulation model; and secondly, we are going to briefly introduce the workloads tested in our simulation.

For our 3D NoC model, our simulation is carried out for a 16-core SNUCA CMP system with shared L2 cache using the Multifacet GEMS simulator [13] based on Simics [14]. To correctly simulate data compression and its effect on NoCs, we have modified the detailed network model of GEMS. Each core has a pair of dedicated instruction/data L1 caches and the L2 cache is divided into 16 banks. The coherence model of caches includes MOESI protocol with 2 distributed on-chip directories implemented on the bottom layer. Directories are used to maintain coherence of memory hierarchies and served as memory controllers; in our simulation, directory entry access costs 6 cycles, same as the L2 cache. So any L2 cache miss at a core will result in a directory access to locate the needed data, which is either in another core's L1 cache or in the main memory. The whole memory address space is interleaved across these two directories, each of which is also a channel to the main memory. The router has a fixed 3-stage pipeline, wormhole switching and 3 virtual channels; the network interface is implemented with a 2-stage pipeline. Compression always consumes one cycle of latency while de-compression takes two cycles.

The simulation parameters also assume each core has 64 KB of L1 cache split for instruction and data. Each L2 cache bank is 256 KB. Three 3D topologies are evaluated. One is having eight cores per die and two stacked dies which forms a 4 by 2 by 2 3D Mesh network. The other two are 4 cores stacked as 4 layers and 2 cores stacked as 8 layers, respectively. They form a 2 by 2 by 4 and a 2 by 1 by 8 3D Mesh topologies, one by another. Note that all planar links for 3D NoCs are 128-bit wide and all vertical links are 16-bit wide. We select these two link widths after considering the footprint of TSVs. Footprint of a TSV is much larger than that of a wire or a driver cell. For example, a typical size of via-last TSVs ranges from 5 to 20 um [5], while that of an inverter cell is only 0.57 um by 2.47 um in the case of OSU's free 45 nm standard cell library. Furthermore, wire-bonding and microbump are believed to be more area hungry according to Ref. [4] as we mentioned in Section 2.

Routers in this 3D NoC model employ deterministic X-Y-Z routing and 2 more ports are needed as connections to routers at neighbor dies/wafers. Packet communication between layers assumes that each 128-bit flit is transferred over 16-bit links in 8 cycles; however, routing and arbitration for vertical going flits are not different from non-vertical going ones.

For simplicity, configurations are summarized in **Table 1**. We use wormhole switching with credit-based flow control for both horizontal and vertical transfers. We assume that the flow con-

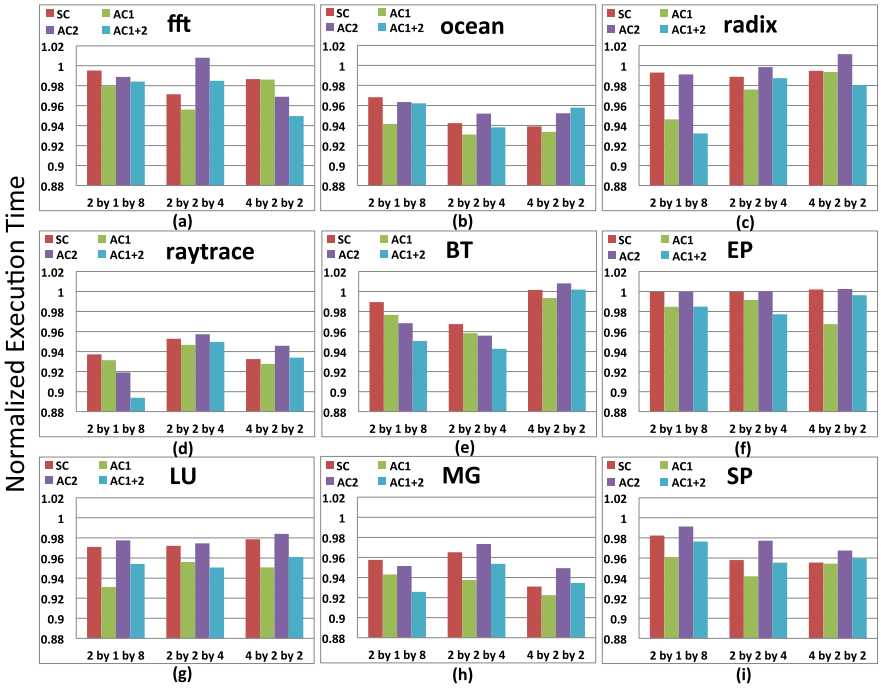

Fig. 9 Normalized execution time with static/adaptive compression on 3D NoCs.

trol signals for vertical are implemented by TSVs, while those for horizontal are implemented with metal wires.

In order to have a diverse performance evaluation, we selected nine workloads from SPLASH-2 [15] and NPB 3 [16] suites for our simulations with 16-core input. The NPB benchmark programs are compiled with OpenMP and problem size of Class S.

5. Results

Depending on memory access characteristics, on-chip bandwidth requirements and compressibility of workload, data compression on 3D NoCs can bring several benefits. In this section, we are going to make clear how these benefits look like in practice. We will discuss and quantify the normalized execution time for 3D NoCs under two compression schemes, static and adaptive. Note that for adaptive compression, we applied each policy separately. In total, there are four sets of results under static compression (SC), compressibility-based compression (AC1), location-based compression (AC2) and the conjunction of AC1 and AC2 (AC1+2). These results are obtained with normalization to execution time under no compression (NC) and they are presented in **Fig. 9** with each histogram representing a workload.

Firstly, static data compression on 3D NoCs is fairly effective. Of the 27 cases (9 workloads with 3 topologies) we have tested, only 4 of them show zero or negative performance improvement, which means for these cases, the overhead of compression is not well covered by the amount of network latency reduced. These 4 cases are, SC of *BT* on 4 by 2 by 2 in Fig. 9 (e) and SCs of *EP* on 2 by 1 by 8, 2 by 2 by 4 and 4 by 2 by 2 in Fig. 9 (f).

Secondly, adaptive control of data compression is more effective than SC. AC1 outperforms SC for all tested workloads and configurations. This is supported by the fact that if compression is beneficial, then AC1 is the same as SC, while if compression is not carried out because of it results in more flits or no flit reduction, then we waste one cycle at the compressibility check, but we save 2 cycles at de-compression and maybe latency at the network. We found the improvement ranges from 1 to 5%, thus avoiding incompressible packets is very useful. As we proposed, AC2 performs better than SC with more layers. With topology of 4 by 2 by 2, AC2 outperforms SC in only one case, between AC2 and SC of *fft* on 4 by 2 by 2 in Fig. 9 (a); but this number climbs up to 6 with topology of 2 by 1 by 8. The 6 cases are, *fft* in Fig. 9 (a), *ocean* in Fig. 9 (b), *radix* in Fig. 9 (c), *raytrace* in Fig. 9 (d), *BT* in Fig. 9 (e) and *MG* in Fig. 9 (h). Similarly, AC1+2 also outperforms SC with more layers; it can also be noted that because of avoiding unnecessary compression which is harmful on layer-crossing traffic, AC1+2 is better than SC for all workloads with topology 2 by 1 by 8.

Thirdly, between AC1 and AC2, AC2 misses chances of compression for traffic travels within layer and it also suffers from unnecessary compression for packets going across layers. For these two reasons, AC1 is generally better than AC2 but with topology of 2 by 1 by 8, it is observed that AC2 outperforms AC1 in two cases as for *raytrace* and *BT*. This means the benefit gained by compressing layer-wise packets with AC1 does not compensate for its compression and de-compression overhead, while AC2's gain from compressing layer-crossing packets well exceeds its unnecessary compression. Another reason is that with more layers, it is less possible for AC2 to lose chances to compress data within a layer.

Finally, after combining the two policies, it is seen that AC1+2 outperforms AC2 in almost all cases with the same reason as AC1 outperforms SC. This means layer-crossing packets also favor the compressibility check, which improves AC2 by denying all incompressible layer-crossing packets. Another important observation is AC1+2 outperforms AC1 in two cases under topology of 4 by 2 by 2, which are *fft* and *radix*. However, this number grows

to 3 for topology of 2 by 2 by 4 with *BT*, *EP* and *LU*; and it further grows to 4 for topology of 2 by 1 by 8 with *radix*, *raytrace*, *BT* and *MG*. We can see AC1+2 also performs better while the chip is implemented with more layers. This is the same as AC2; if having more layers, AC1+2 also loses less chances of traffic within a layer.

More specifically, for both 2 by 2 by 4 and 4 by 2 by 2, AC1 has been recorded a performance improvement of up to 7% over NC, and is better than SC, AC2 and AC1+2. This 7% of improvement with AC1 is seen in Fig. 9 (b), (d) and (h) for *ocean* on 2 by 2 by 4, *raytrace* on 4 by 2 by 2 and *MG* on 4 by 2 by 2. However, with 2 by 1 by 8, AC1+2 is seen to have the best performance improvement of up to 11% over NC in Fig. 9 (d) for *raytrace*.

6. Related Work

In this section, we present a short summary of previous work related to this paper. Data compression for NoCs, as an efficient on-chip optimization, has been extensively studied for 2D design [10], [11], [12]. In Ref. [10], the authors were the first to apply frequent pattern compression on a CMP with Network-on-Chip architecture. Their primary goal was to make a comparison between cache compression and network compression with the same algorithm, in terms of their effects on performance and energy consumption. Both Refs. [11] and [12] were about compressing data on NoCs with another candidate algorithm, frequent value compression. Although their results are showing positive feedback, we believe that for any architecture having multiple communicating nodes, frequent value compression can be inefficient because of its overheads of area and synchronization make it scale poorly. In Ref. [11], the authors also propose a solution to the area overhead and an adaptive compression control mechanism taking into account the network congestion.

Before the study of data compression on NoCs was carried out, there were already many efforts of applying data compression on bus and cache [6], [7], [8], [9]. Moreover, a study carried out in Ref. [17] had proved that both cache and bus compression are highly efficient in terms of further scaling CMP designs.

7. Conclusions

In this paper, we have evaluated how adaptive data compression affects system performance for CMPs implemented with 3D NoCs. We also presented what difference on performance is made with adaptive schemes of data compression proposed in the paper. We find that in a bandwidth limited situation like a CMP with 3D NoCs having multiple connected layers, adaptive data compression with location-based control or with both compressibility and location based control is very promising if the number of layers continues to grow.

Furthermore, according to the evaluation result, we believe that if frequent pattern compression is to be utilized, then compressibility check is a must since it is always better than static compression. Secondly, if a 3D implementation has many layers and few cores per layer, AC1+2 is very efficient since it targets specifically at the vertical bandwidth limitation and most of the traffic are layer-crossing. Finally, although the improvements vary case by case, we believe our results are quite conservative since we simulate with Simics whose processor model is in-order and we use a relatively smaller problem size for our workloads (especially the NPB ones). In practice, modern processor cores are generally more advanced with a higher bandwidth requirement. In consequence, we believe that a more promising improvement than our results can be expected if a similar 3D design has our adaptive FPC implemented.

Reference

[1] Black, B., Annavaram, M., Brekelbaum, N., DeVale, J., Jiang, L., Loh, G.H., McCaule, D., Morrow, P., Nelson, D.W., Pantuso, D., Reed, P., Rupley, J., Shankar, S., Shen, J.P. and Webb, C.: Die Stacking (3D) Microarchitecture, *Proc. International Symposium on Microarchitecture* (*MICRO'06*), pp.469–479 (2006).

[2] Kumagai, K., Yang, C., Goto, S., Ikenaga, T., Mabuchi, Y. and Yoshida, K.: System-in-Silicon Architecture and its Application to H.264/AVC Motion Estimation for 1080HDTV, *Proc. International Solid-State Circuits Conference* (*ISSCC'06*), pp.430–431 (2006).

[3] Burns, J., McIlrath, L., Keast, C., Lewis, C., Loomis, A., Warner, K. and Wyatt, P.: Three-Dimensional Integrated Circuits for Low-Power High-Bandwidth Systems on a Chip, *Proc. International Solid-State Circuits Conference* (*ISSCC'01*), pp.268–269 (2001).

[4] Davis, W.R., Wilson, J., Mick, S., Xu, J., Hua, H., Mineo, C., Sule, A.M., Steer, M. and Franzon, P.D.: Demystifying 3D ICs: The Pros and Cons of Going Vertical, *IEEE Design and Test of Computers*, Vol.22, No.6, pp.498–510 (2005).

[5] Kim, D.H., Athikulwongse, K. and Lim, S.K.: A Study of Through-Silicon-Via Impact on the 3D Stacked IC Layout, *Proc. IEEE/ACM International Conference on Computer-Aided Design* (*ICCAD'09*), pp.674–680 (2009).

[6] Alameldeen, A.R. and Wood, D.A.: Frequent Pattern Compression: A Significance-Based Compression Scheme for L2 Caches, Technical Report 1500, Computer Sciences Department, University of Wisconsin-Madison (2004).

[7] Alameldeen, A.R.: Using Compression to Improve Chip Multiprocessor Performance, PhD Thesis, University of Wisconsin at Madison (2006).

[8] Alameldeen, A.R. and Wood, D.A.: Adaptive Cache Compression for High-Performance Processors, *ACM SIGARCH Computer Architecture News*, Vol.32, No.2, pp.212–223 (2004).

[9] Thuresson, M., Spracklen, L. and Stenstrom, P.: Memory-Link Compression Schemes: A Value Locality Perspective, *IEEE Trans. Comput.*, Vol.57, No.7, pp.916–927 (2008).

[10] Das, R., Mishra, A.K., Nicopoulos, C., Park, D., Narayanan, V., Iyer, R., Yousif, M.S. and Das, C.R.: Performance and Power Optimization through Data Compression in Network-on-Chip Architectures, *Proc. IEEE International Symposium on High Performance Computer Architecture* (*HPCA'08*), pp.215–225 (2008).

[11] Jin, Y., Yum, K.H. and Kim, E.J.: Adaptive Data Compression for High-Performance Low-Power On-Chip Networks, *Proc. IEEE/ACM International Symposium on Microarchitecture* (*MICRO'08*), pp.354–363 (2008).

[12] Zhou, P., Zhao, B., Du, Y., Xu, Y., Zhang, Y., Yang, J. and Zhao, L.: Frequent Value Compression in Packet-based NoC Architectures, *Proc. Asia and South Pacific Design Automation Conference* (*ASP-DAC'09*), pp.13–18 (2009).

[13] Martin, M.M.K., Sorin, D.J., Beckmann, B.M., Marty, M.R., Xu, M., Alameldeen, A.R., Moore, K.E., Hill, M.D. and Wood, D.A.: Multifacets General Execution-driven Multiprocessor Simulator (GEMS) Toolset, *ACM SIGARCH Computer Architecture News*, Vol.33, No.4, pp.92–99 (2005).

[14] Magnusson, P.S., Christensson, M., Eskilson, J., Forsgren, D., Hallberg, G., Hogberg, J., Larsson, F., Moestedt, A. and Werner, B.: Simics: A Full System Simulation Platform, *IEEE Computer*, Vol.35, No.2, pp.50–58 (2002).

[15] Singh, J.P., Weber, W. and Gupta, A.: SPLASH: Stanford Parallel Applications for Shared-Memory, *ACM SIGARCH Computer Architecture News*, Vol.20, No.1, pp.5–44 (1992).

[16] Jin, H., Frumkin, M. and Yan, J.: The OpenMP Implementation of NAS Parallel Benchmarks and Its Performane, NAS Technical Report NAS-99-011, NASA Advanced Supercomputing (NAS) Division (1999).

[17] Rogers, B., Krishna, A., Bell, G., Vu, K., Jiang, X. and Solihin, Y.: Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling, *Proc. International Symposium on Computer Architecture* (*ISCA'09*), pp.371–382 (2009).

[18] Wentzlaff, D., Griffin, P., Hoffmann, H., Bao, L., Edwards, B., Ramey,

C., Mattina, M., Miao, C.-C., Brown III, J.F. and Agarwal, A.: On-Chip Interconnection Architecture of the Tile Processor, *IEEE Micro*, Vol.27, No.5, pp.15–31 (2007).
[19] Vangal, S.R., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Singh, A., Jacob, T., Jain, S., Erraguntla, V., Roberts, C., Hoskote, Y., Borkar, N. and Borkar, S.: An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS, *IEEE Journal of Solid-State Circuits*, Vol.43, No.1, pp.29–41 (2008).
[20] Benini, L. and De Micheli, G.: *Networks on Chips: Technology And Tools*, Morgan Kaufmann (2006).
[21] Sheibanyrad, A., Petrot, F. and Janstch, A.: *3D Integration for NoC-Based SoC Architectures*, Springer (2010).
[22] Pavlidis, V.F. and Friedman, E.G.: 3-D Topologies for Networks-on-Chip, *IEEE Trans. Very Large Scale Integration Systems*, Vol.15, No.10, pp.1081–1090 (2007).
[23] Kim, J., Nicopoulos, C., Park, D., Das, R., Xie, Y., Vijaykrishnan, N., Yousif, M. and Das, C.: A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures, *Proc. International Symposium on Computer Architecture* (*ISCA'07*), pp.138–149 (2007).
[24] Park, D., Eachempati, S., Das, R., Mishra, A.K., Narayanan, V., Xie, Y. and Das, C.R.: MIRA: A Multi-layered On-Chip Interconnect Router Architecture, *Proc. International Symposium on Computer Architecture* (*ISCA'08*), pp.251–261 (2008).
[25] Ramanujam, R.S. and Lin, B.: Randomized Partially-Minimal Routing on Three-Dimensional Mesh Networks, *IEEE Computer Architecture Letters*, Vol.7, No.2, pp.37–40 (2008).

Yuan He received his B.Sc. and M.E. (Hons) from the University of Auckland, New Zealand in 2005 and 2009, respectively. He is currently a Ph.D. student with the University of Tokyo, Japan. His research is focused mainly on optimizations for interconnection networks of multi-core processors. He is a student member of IEEE and IPSJ.

Hiroki Matsutani received the B.A., M.E., and Ph.D. degrees from Keio University in 2004, 2006, and 2008, respectively. He was a Research Fellow with Graduate School of Information Science and Technology, the University of Tokyo, from 2009 to 2011. He is currently an Assistant Professor at Department of Information and Computer Science, Keio University. His research interests include the areas of computer architecture and interconnection networks.

Hiroshi Sasaki received his B.E., M.E., and Ph.D. degrees from the University of Tokyo in 2003, 2005, and 2008, respectively. He was a Project Assistant Professor at the University of Tokyo from 2008 to 2011, and is currently a Project Associate Professor at Department of Advanced Information Technology, Kyushu University. His research interests include computer architecture and operating systems for future microprocessors.

Hiroshi Nakamura received his B.E., M.E., and Ph.D. degrees in Electrical Engineering from the University of Tokyo in 1985, 1987, and 1990, respectively. He was a Visiting Associate Professor at the University of California, Irvine from 1996 to 1997. He is currently a Professor of Department of Information Physics and Computing at the University of Tokyo. His research interests include low-power processor, VLSI design, power-aware computing, high-performance computer systems, and dependable computing. He is a member of IEICE and IPSJ, and a senior member of IEEE and ACM.