

# Explorations of Optimal Core and Cache Placements for Chip Multiprocessor

Thomas Canhao Xu, Pasi Liljeberg, Hannu Tenhunen

Department of Information Technology, University of Turku, 20014 Turku, Finland  
Turku Center for Computer Science (TUCS), Joukahaisenkatu 3-5 B, 20520 Turku, Finland  
canxu, pasi.liljeberg, hannu.tenhunen@utu.fi

**Abstract**—In this paper, we study and analyse optimal core and cache placements for modern Chip Multiprocessors (CMPs). As the number of cores increases, traditional on-chip interconnects such as bus and crossbar suffer from poor scalability and low efficiency. Ring based design has been proposed and implemented to mitigate these problems. However, the continuation growth of number of cores will render the ring interconnect infeasible. Network based designs are therefore proposed for future CMPs for better scalability. In this paper, we explore the interconnect of a state-of-the-art CMP. We analyse and compare the implementation of the ring-based and the network-based interconnect. The placement of cores and caches in a network is proved crucial for system performance. We investigate optimal core/cache placement for CMPs. The benchmark results are presented by using a cycle accurate full system simulator. Results show that, by using the optimal network interconnect, compared with the ring interconnect, the average network latency and execution time are reduced by 11.93% and 19.53% respectively, for four configurations and two applications.

## I. INTRODUCTION

We have witnessed multi-core processors emerging with increasing number of cores and complex on-chip interconnect in the past few years. The constraints of chip clock frequency, power consumption and heat dissipation have pushed the chip designers to integrate multiple cores rather than to improve single-core performance. A prevalent approach to increase chip performance is to add more cores. Normally, each core has its own computational units, registers and private caches. A last level cache will be shared by all cores. Intel Pentium-D<sup>1</sup> is one of the first CMP which embedded two dies on a processor package. Nowadays, quad-core CMPs such as Intel Core i5, i7 [1] and AMD Phenom II [2] are mainstream for desktop systems. The integration of more cores on a CMP is under intensive research. For example, AMD has announced its twelve-core Opteron CMP with two dies on a chip, each of which has six cores [3]. The newest Xeon, based on Westmere-EX architecture, has been fabricated by Intel with ten cores [4]. Sun and IBM have their own CMP design as well [5], [6]. It is predictable that in the future, the integration of more cores on a single chip will move on.

The current communication schemes in CMPs are mainly based on the shared bus and crossbar architectures. In small scale CMPs, crossbar based architectures are advantageous

since they have a simple topology and are easy to implement. Besides, the latencies are usually fixed and messages can be broadcasted. However, as the system size increases, crossbar based communication architectures are no longer feasible due to the circuit complexity and energy efficiency. For example, crossbar interconnect is used in Intel Nehalem and Westmere architectures. Each core has its own data path (around 1,000 wires) to the central interconnect. This provides high bandwidth for a small number of agents (e.g. cores, caches and memory controllers). The latency grows as the number of agents grows, e.g. the latency to the last level cache with six cores is 22% higher than that in four cores [7]. The crossbar has to be redesigned and it becomes more complex. Another research show that, high-radix crossbars are not feasible due to the area overhead and the frequency limitations [8]. Obviously, we cannot design a crossbar based CMP by simply adding more cores/caches.

Due to the limited scalability of crossbar switches, academics and industries have moved to new approaches. Sun SPARC T3 applied a hierarchical crossbar to alleviate the scaling problem. While IBM Power 7, Intel Sandy Bridge, Nehalem-EX and Westmere-EX rely on a ring interconnect. Previous researches have focused on the performance and efficiency of crossbar/ring/mesh interconnects. For example, Cheolmin Park et al. [9] presented a 1.2 TB/s ring interconnect implementation, providing on-chip communication for 8 cores, 8-port parallel-access 24 MB cache, and 2 system interfaces. Hyper ring, hierarchical ring and hybrid ring have been proposed to improve efficiency further [10], [11]. Jesus Camacho Villanueva et al. [12] compared the performance differences between crossbar, ring and mesh interconnects. In this paper, we investigate the placement of cores and caches in a 2D mesh network interconnect. We model a CMP based on state-of-the-art processors. We analyze the impact of different placements of cores/caches in a mesh network. Optimal placements of several configurations have been found and discussed. We present the performance using a full system simulator. To the best of our knowledge, this is the first paper that analyzes different core/cache placements in a mesh network.

## II. MODERN CHIP MULTIPROCESSOR DESIGN

As aforementioned, there have been different interconnect topologies for CMPs. Conventional designs are focused on crossbar or hierarchical crossbar. While recently, designers

<sup>1</sup>Intel and Pentium are trademarks or registered trademarks of Intel or its subsidiaries. Other names and brands may be claimed as the property of others.

TABLE I: Comparison of different Chip Multiprocessors.

| CMP                   | Cores    | Last Level Cache | Number of Transistors | Process | Die Size                                   | On-chip Interconnect  |
|-----------------------|----------|------------------|-----------------------|---------|--------------------------------------------|-----------------------|
| AMD MagnyCours [3]    | 12 (2×6) | 12MB (2×6MB)     | 1,808M (2×904M)       | 45nm    | 692mm <sup>2</sup> (2×346mm <sup>2</sup> ) | Crossbar              |
| IBM Power 7 [6]       | 8        | 32MB             | 1,200M                | 45nm    | 567mm <sup>2</sup>                         | Ring                  |
| Intel Nehalem [1]     | 4        | 8MB              | 731M                  | 45nm    | 263mm <sup>2</sup>                         | Crossbar              |
| Intel Westmere [1]    | 6        | 12MB             | 1,170M                | 32nm    | 239mm <sup>2</sup>                         | Crossbar              |
| Intel SandyBridge [4] | 4        | 8MB              | 995M                  | 32nm    | 216mm <sup>2</sup>                         | Ring                  |
| Intel Nehalem-EX [1]  | 8        | 24MB             | 2,300M                | 45nm    | 684mm <sup>2</sup>                         | Ring                  |
| Intel Westmere-EX [4] | 10       | 30MB             | 2,600M                | 32nm    | 513mm <sup>2</sup>                         | Ring                  |
| Sun SPARC T3 [5]      | 16       | 6MB              | 1,000M                | 40nm    | 377mm <sup>2</sup>                         | Hierarchical Crossbar |

are moving from crossbar to ring interconnect to improve scalability. Ring has natural ordering properties, making design, validation and routing simpler. Table I shows the key features of state-of-the-art CMPs. Take the Westmere-EX for example (Figure 1), it uses a bi-directional ring to connect the last level caches to the cores. The ring consists of 4 different rings, namely request, acknowledge, snoop and data. The data ring is pipelined and uses 32 bytes data width. For each processor cycle, data can move one hop on the ring for either directions. These 4 rings are responsible for the cache/memory coherency. It is reported that [13], although the sizes of last level cache in Nehalem and Sandy Bridge are the same (8MB), the access latency is significantly reduced from 35 cycles in Nehalem (Minimal latency is 35 cycles if the frequencies of core and uncure are the same) to 26-31 cycles in Sandy Bridge (Depending on the hop count to the cache slices). Several factors contribute to the reduced latency. For example, the cache architecture of Nehalem is uniform and divided into multiple levels. The access latency of the 8MB last level cache is determined by the longest access latency to the furthest data arrays and tags. While in Sandy Bridge, the 8MB cache is split into 4 slices, each 2MB. The access latencies to access the cache slices are decreased. Each slices has different access latency, making it a non-uniform cache architecture.

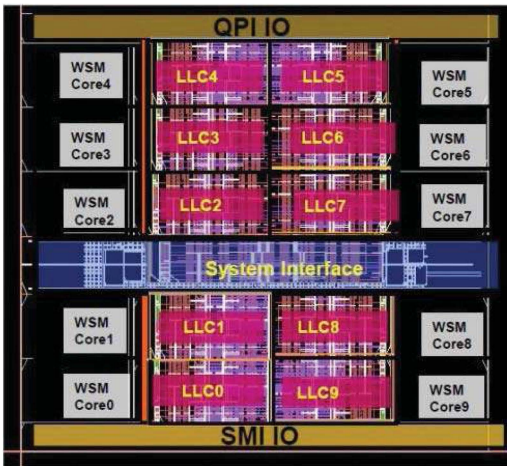


Fig. 1: Intel Westmere-EX architecture, presented by Intel at Hot Chips 2010.

The scaling of the ring has limitations. As data travel across the ring, it can block accesses from other agents. With each additional agent, the available bandwidth will be reduced and

the average hop count will increase. In this case, network-based design has been introduced. The concept of computer network is proposed for on-chip interconnect [14]. A large number of components can be integrated on a single chip with much better scalability compared with crossbar and ring [14]. Intel has demonstrated two chips with network interconnect: a special-purpose chip with 80 core, 100M transistor, 275mm<sup>2</sup> under 65nm processing technology [15], and a general-purpose x86 chip with 48 cores [16]. Figure 2 shows a homogeneous Network-on-Chip (NoC) with 4×4 mesh (16 nodes). The underlying network is comprised of network links and routers (R), each of which is connected to a processor core (Core/L1\$) or/and cache (L2\$) via a network interface (NI). The basic architectural unit of a NoC is the tile/node (N) which consists of a router, its attached NI and core/cache, and the corresponding links. Communication among agents is achieved via network packets. It is noteworthy that each node can include different component, e.g. a node with only Core/L1\$, and a node with only L2\$, creating a heterogeneous network.

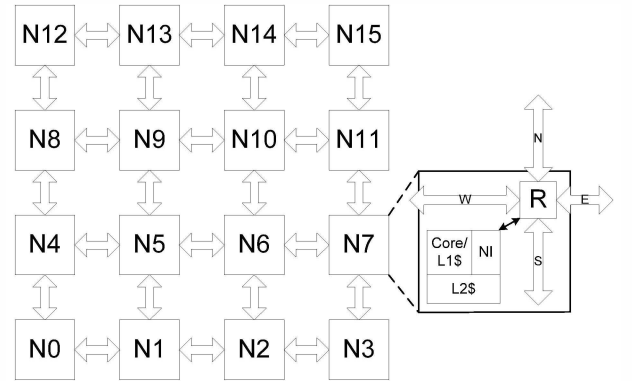


Fig. 2: An on-chip network of 4×4 using mesh topology.

### III. PLACEMENT OF CORE AND CACHE

In this section, we analyse the area consumption of cores and caches, as well as the performance with different placements of cores and caches.

#### A. Area of core and cache

As is shown in Figure 1, cores and caches in the Westmere-EX have similar area. For example, a 32nm Westmere core has an area of about 17mm<sup>2</sup>. The last level cache in Westmere-EX is split into 10 slices, each 3MB. Notice that each slice is about the same size as a core. In our paper, we assume that:

- The area of a core and a cache slice are the same,
- The last level cache consists of multiple slices, the number of cores and cache slices is equal.

In this case, an 8-core CMP has 8 cores and 8 cache slices. These assumption fits the design principles of Intel and IBM [1], [4], [6].

### B. Ring and network connection

In a ring connection, the average hop count accessing cache slices for all processors are the same. For example, Figure 3a shows an abstract model of 8-core Nehalem-EX. The cores (C0 to C7) are connected to the central bi-directional ring (the connections between cores and ring are not shown in figure). Figure 3b and 3c demonstrate the hop counts required for accessing each cache slice of C0 and C1, respectively. The latencies for accessing each cache slice is different. Considering a perfect application with average access to cache slices, for the 8-slice cache, each slice has an access probability of  $1/8$ . Real-world applications are most likely to have different cache access patterns, i.e. the access frequency to certain slices might be higher than others. In this case, the addresses of data kept in the cache arrays can be distributed among the cache slices with a uniform distributed hash function [13]. However, a system with billions of cache accesses tend to have a balanced access to all slices. We define  $AHP_{Cn}$  as the Average Hop Count accessing all cache slices for a core  $Cn$ :

$$AHP_{Cn} = \frac{\sum_{i=0}^m \text{Distance}(C_n, \text{Cache}_i)}{m+1}$$

Such that:  $m+1 = (\text{Number of cache slices})$

Notice that the average hop counts for C0 ( $AHP_{C0}$ ) and C1 ( $AHP_{C1}$ ) to all cache slices are the same, which is 3. We define  $AHP_{All}$  as the average value of all  $AHP_{Cn}$ :

$$AHP_{All} = \frac{\sum_{i=0}^n AHP_{Ci}}{n+1}$$

Obviously, the  $AHP_{All}$  for Figure 3a is 3 since all cores have the same average latency to the cache.

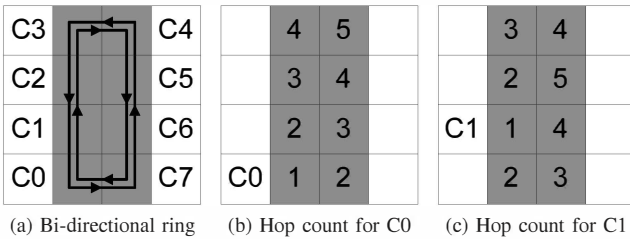


Fig. 3: An example of 8-core CMP, with Bi-directional ring interconnect (a, Cx as Core x), and access latency (shown as hop count) for Core 0 (b) and Core 1 (c) of each cache slice.

By implementing a network interconnect, the  $AHP$ s can be non-uniform for different cores. Manhattan Distance, or MD, is used to calculate the hop count in a network-based interconnect. For example,  $AHP_{C0}$  in a network is the same as in a ring, however  $AHP_{C1}$  in a network is just 2.5 compared with 3 in a ring. We calculate that in Figure 3a, the  $AHP_{All}$

for a network interconnect is 2.75, 8.3% lower than that in a ring.

### C. Placements of Caches

The different placement of cache slices can have a dramatic impact on the performance of the CMP. Figure 4a shows a worst case placement of 8 cache slices. In this configuration, all the cache slices are placed on one side of the chip (gray nodes), causing high delays for the cores in the other side of the chip ( $AHP_{All}=3.25$ ).

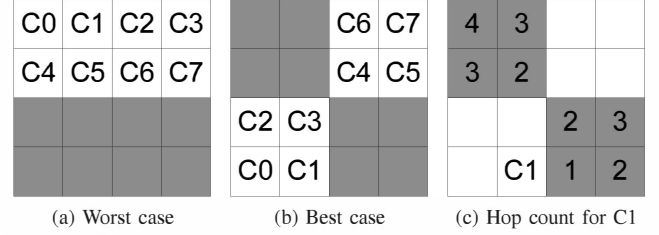


Fig. 4: Examples of worst case placement (a), best case placement (b), and access latency Core 1 (c) of each cache slice of the best case.

To minimize cache access latencies for all cores, the placement of cache slices should be considered carefully. For example, an 8-core CMP with 8 cache slices has  $\binom{16}{8}=12,870$  different cache placement possibilities. The value of  $AHP_{All}$  varies significantly with different placements. We explore the placements by writing a program that exhaustively enumerate all possibilities and output the  $AHP_{All}$  (Algorithm 1). The outputs are then processed and sorted for analysis. A best case placement of 8 cache slices is illustrated in Figure 4b. We observe that there are 90 placements with lowest  $AHP_{All}$  (2.5), and 4 placements with highest  $AHP_{All}$  (3.25). The cache access latencies of Core 1 are demonstrated in Figure 4c. It is noteworthy that  $AHP_{C3}$  and  $AHP_{C4}$  are lowest among 8 cores, while  $AHP_{C0}$  and  $AHP_{C7}$  are highest.

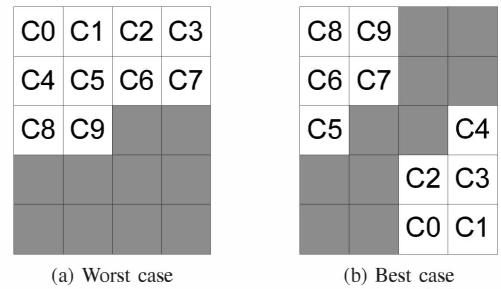


Fig. 5: Examples of worst case placement (a), and best case placement (b), for a 10-core 10-cache configuration.

We continue our investigation for a 10-core CMP, similar to Westmere-EX. In this case, there are  $\binom{20}{10}=184,756$  different cache placement possibilities. The simulation output has 4 worst case placements and 1,240 best case placements. Figure 5a and 5b show two of these. The  $AHP_{All}$  for worst case placements and best case placements are 3.68 and 2.86 respectively, while the value is 3.5 in a ring interconnect.

**Algorithm 1** The pseudo code of calculating  $AHP_{All}$ , for a combination of cache slices placement.

```

void calculate_ahpall()
{
    for(i=0; i<Combinations; i++) //Calculate for all
        combinations
    {
        for(j=0; j<C_Num; j++) C[j]=C_Comb[i*C_Num+j];
        //C_Comb is an array of
        //all cache placement
        //combinations

        last_node=0;
        for(k=0; k<Net_x*Net_y-C_Num; k++) //Calculate for all
            caches in the CMP
        {
            Cores[k]=last_node; //Cores is set of cores
            for(l=0; l<Net_x*Net_y; l++)
            {
                for(m=0; m<C_Num; m++)
                {
                    if(Cores[k]==C_Comb[m]) Cores[k]++;
                    if(k>0&&(Cores[k-1]==Cores[k])) Cores[k]++;
                }
                last_node=Cores[k];
            }

            for(x=0; x<Net_x*Net_y-C_Num; x++) //for all cores
            {
                for(y=0; y<C_Num; y++) //for all caches
                {
                    tmp+=MD(Cores[x], C_Comb[y]); //Calculate the MD
                    //from a Core to a
                    //cache slice
                }
                avg_dist=tmp/C_Num;
                total_dist+=avg_dist;
                avg_dist=0;
                tmp=0;
            }
            final_ahpall[i]=total_dist/(Net_x*Net_y-C_Num);
            //Final AHP_all for
            //a combination
        }
        total_dist=0;
    }
}

```

The best case  $AHP_{All}$  for a 12-core CMP is similar to the 8-core, where 6 cores are placed on one side, and the other 6 cores are placed on the other side of the CMP (Figure 6a). The  $AHP_{All}$  for worst case placements and best case placements are 3.19 and 4.25 respectively, while the value is 4 in a ring interconnect. We find 1,860 best case placements from 2,704,156 combinations. Figure 6b shows a best case placement for a 14-core CMP, notice that the placement of cores and caches is similar to the 10-core configuration.

#### D. Discussion

To compare the difference between ring and network interconnect, we summarize and calculate the  $AHP_{All}$  for several configurations. Table II shows the data (NW for worst case placement of a network, NB for best case placement of a network, NB%Ring for percentage of NB compared with Ring, NB%NW for percentage of NB compared with NW). It is shown that, on average, the  $AHP_{All}$  for a best placement of cache slices in a network interconnect is around 25% less

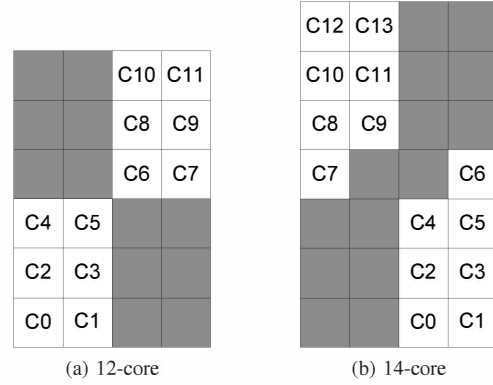


Fig. 6: Best case placements for a 12-core CMP (a), and 14-core CMP (b).

than that in a ring. Researches have shown that [13], due to the implementation of ring interconnect, the last level cache hit latency has reduced from 35-40 cycles (Nehalem) to 26-31 cycles (Sandy Bridge).

TABLE II: Comparison of  $AHP_{All}$  for different interconnect and placement

| Configuration    | Ring | NW   | NB   | NB%Ring | NB%NW |
|------------------|------|------|------|---------|-------|
| 8 Core/8 Cache   | 3.00 | 3.25 | 2.50 | 76.9%   | 83.3% |
| 10 Core/10 Cache | 3.50 | 3.68 | 2.86 | 81.7%   | 77.7% |
| 12 Core/12 Cache | 4.00 | 4.25 | 3.19 | 79.9%   | 75.2% |
| 14 Core/14 Cache | 4.50 | 4.69 | 3.54 | 78.7%   | 75.5% |

We use abstract models of  $4 \times N$ , where  $N$  equals to half of the number of cores. For example, we use  $4 \times N$  network in "10 Core/10 Cache" configuration. This model fits well with smaller number of cores/caches. According to [17], a model shape has lower number of  $AHP$  (this  $AHP$  is for all nodes, assuming a homogeneous model) if it is closer to a square. Their conclusion applies to our heterogeneous model as well. Considering the "12 Core/12 Cache" configuration which has 24 nodes, the communication overhead for a  $4 \times 6$  model is smaller than that in  $3 \times 8$  and  $2 \times 12$ . In this case, we don't consider models such as  $3 \times 8$  and  $2 \times 12$ . CMPs with a higher number of cores are not considered in our paper, due to the fact that the number of cores and cache slices may not be equal in the future. For instance, current desktop CMP design trends tend to have smaller and faster caches for better user interface experience, while a CMP designed for server environment have larger caches.

However, it is hard to define a rule of thumb for optimal placement of cores/caches. We note that exhaustive simulation is feasible for a smaller number of nodes, e.g. a  $6 \times 6$  model with 18 cores and 18 cache slices has  $\binom{36}{18} = 9,075,135,300$  different placement possibilities, a  $5 \times 8$  model with 20 cores and 20 cache slices has  $\binom{40}{20} = 137,846,528,820$  possibilities. The design space grows exponentially as the number of cores increases, making the exhaustive method to be unpractical. Researches have demonstrated that [18], divide and conquer

method can be used for a huge design space, however this method only applies with certain limitations. Heuristic-guided methods can reduce the computational complexity from a large search space, notwithstanding the non-optimal outputs [19].

#### IV. EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation under different CMP configurations. Applications are selected from SPLASH-2 [20] and PARSEC [21].

##### A. Experiment Setup

The simulation platform is based on a cycle-accurate full system simulator which can produce detailed evaluation results. Shortest distance routing is used in the ring interconnect, in which the direction is chosen based on the distance from the source to the destination. Deterministic X-Y routing algorithm has been selected in the network interconnect to avoid deadlocks, in which a flit is first routed to the X direction and last to the Y direction. We assume the latency of the switches is one cycle, both for the ring and network. The link latency for any neighbouring nodes is one cycle as well.

TABLE III: System Configuration Parameters

| Processor configuration      |                                                                                                                     |
|------------------------------|---------------------------------------------------------------------------------------------------------------------|
| Instruction Set Architecture | UltraSPARC III+                                                                                                     |
| Number of Processors         | 8, 10, 12 and 14                                                                                                    |
| Core and Network Frequency   | 2GHz                                                                                                                |
| Cache Configuration          |                                                                                                                     |
| L1 Cache                     | Private, split instruction and data cache, each cache is 32KB. 4-way associative, 64-Byte line, 3-cycle access time |
| L2 Cache                     | Shared, unified 24MB, 30MB, 36MB and 42MB (8, 10, 12 and 14 banks, each 3MB). 64-Byte line, 10-cycle access time    |
| Cache Coherence Protocol     | MOESI                                                                                                               |
| Cache Hierarchy              | SNUCA                                                                                                               |
| Memory Configuration         |                                                                                                                     |
| Size                         | 4GB DRAM                                                                                                            |
| Access Latency               | 160 cycles                                                                                                          |
| Requests per Processor       | 16 outstanding                                                                                                      |
| Network Configuration        |                                                                                                                     |
| Router Scheme                | Wormhole                                                                                                            |
| Flit Size                    | 128 bits                                                                                                            |

We use ring and network interconnects to model a single-chip CMP with 8, 10, 12 and 14 cores (16, 20, 24 and 28 nodes). A full system simulation environment has been implemented. The simulations are run on the Solaris 9 operating system based on UltraSPARC III+ instruction set in-order issue structure. Each processor core has a private write-back L1 cache. The L2 cache shared by all processors is split into slices/banks. The size of each cache bank is 3MB; hence the total sizes of the shared cache are 24MB, 30MB, 36MB and 42MB respectively. The simulated memory/cache architecture mimics SNUCA [22]. A two-level distributed directory cache coherence protocol called MOESI based on MESI [23] has been implemented in our memory hierarchy in which each L2 bank has its own directory. The protocol has five types of cache line status: Modified (M), Owned (O), Exclusive

(E), Shared (S) and Invalid (I). We use Simics [24] full system simulator as our simulation platform. The detailed configurations of processor, cache and memory can be found in Table III. Workloads used in this paper include FFT from SPLASH-2 and swaptions from PARSEC. The FFT is a one-dimensional, radix-n, six-step algorithm, which is optimized to minimize inter processor communication. The communication between processors only take place at the last stage of the execution. However the coherence traffic and cache miss rate are very high. Swaptions is a workload that employs Monte Carlo simulation to compute prices.

##### B. Result Analysis

We evaluate performance in terms of Average Network Latency (ANL) and execution time. ANL represents the average number of cycles required for the transmission of all cache/memory messages. The number of cycles of each message is calculated as, from the injection of the message header into the on-chip interconnect at the source node, to the reception of the tail flit at the destination node. Under the same configuration and workload, lower latency and execution time are favourable. The results are illustrated in Figures 7 and 8.

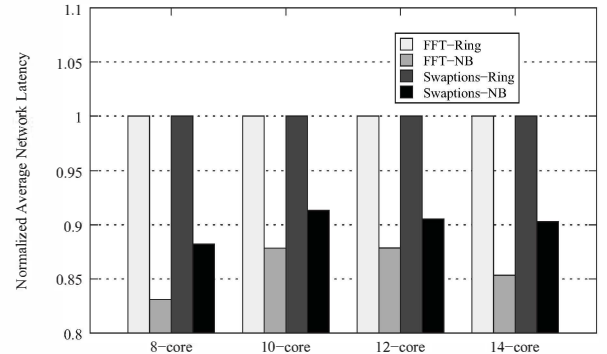


Fig. 7: Normalized average network latency in cycles.

Figure 7 shows that, on the whole, the average network latency of our optimal core/cache placement in a network interconnect (\*-NB in the figure) outperforms the ring interconnect (\*-Ring in the figure) in four configurations. The improvement is more significant in the 8-core configuration, with 16.89% reduced latency in FFT and 11.78% reduced latency in Swaptions, respectively. This is primarily due to the fact that the  $AHP_{All}$  for NB is much lower than that in the Ring. We notice that, the on-chip communication consist of cache coherent messages, cache data and memory control/data messages. Out of these messages, the cache related control/data messages are the dominant (over 90%). In this case, lower average access latency to the cache slices will transfer into lower average network latency. We also note, the improvement is more significant in application with higher cache access frequencies, for example FFT. Overall, the average network latency has reduced 11.93% for four configurations.

As depicted in Figure 8, in terms of execution time, the optimal core/cache placement has considerable advantage

compared with the ring. Overall, the reduction of execution time is more significant than that in the average network latency. Comparing with the ring interconnect, on average, the execution time for our proposed core/cache placement is reduced by 19.53% in four configurations. Again, the higher cache/memory traffic in FFT benefits more from reduced  $AHP_{All}$ , compared with Swaptions. Results shown that, the execution time of FFT has reduced 22.62% in four configurations, while the reduction in Swaptions is 16.43%. This proves that FFT is more sensitive to core/cache latency than Swaptions as well. The savings of  $AHP_{All}$  have translated directly into the reduced execution time.

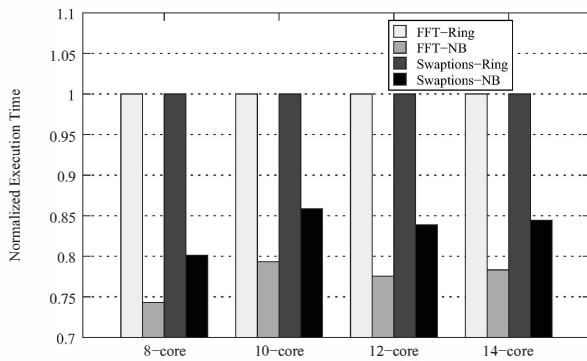


Fig. 8: Normalized execution time.

## V. CONCLUSION

We investigated the optimal placement of cores and caches in a network-based interconnect for near future CMPs in this paper. The crossbar and ring interconnect could be a major bottleneck of future CMP design. This bottleneck could be alleviated by network interconnect. However, we found that the placement of cores and caches in mesh networks have a significant impact to the system performance. Hence, we explored the optimal placements of cores and caches. To prove the feasibility of our proposed optimal architecture, we simulated four CMP configurations, using applications from SPLASH-2 and PARSEC. Experimental results shown that, in general, the average network latency and execution times are reduced by 11.93% and 19.53% respectively. Our study provided a direction for designing future CMPs.

## ACKNOWLEDGMENT

This work is supported by Academy of Finland, Nokia Foundation and Finnish Foundation for Technology Promotion (Tekniikan edistämissäätiö). The authors would like to thank the anonymous reviewers for their feedback and suggestions.

## REFERENCES

- [1] Intel, "Intel core i7 processor extreme edition and intel core i7 processor datasheet, volume 1," July 2011, <http://download.intel.com/design/processor/datashts/320834.pdf>.
- [2] AMD, "Family 10h amd phenom processor product data sheet," July 2011, [http://www.amd.com/us-en/assets/content\\_type/white\\_papers\\_and\\_tech\\_docs/44109.pdf](http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/44109.pdf).
- [3] AMD, "The amd opteron 6000 series platform," July 2011, <http://www.amd.com/us/products/server/processors/6000-series-platform/pages/6000-series-platform.aspx>.

- [4] Intel, "Intel xeon processor e7-8870," July 2011, <http://ark.intel.com/Product.aspx?id=53580>.
- [5] M. Tremblay and S. Chaudhry, "A third-generation 65nm 16-core 32-thread plus 32-scout-thread cmt sparc processor," in *ISSCC 2008*, February 2008, pp. 82–83.
- [6] IBM, "Ibm power 7 processor," in *Hot chips 2009*, August 2009.
- [7] S. Wasson, "Intel's xeon 5600 processors, westmere-ep adds two more cores to an already-potent mix," July 2010, <http://techreport.com/articles.x/19196/4>.
- [8] A. Pullini, F. Angiolini, S. Murali, D. Atienza, G. De Micheli, and L. Benini, "Bringing nocs to 65 nm," *Micro, IEEE*, vol. 27, no. 5, pp. 75–85, sept.-oct. 2007.
- [9] C. Park, R. Badeau, L. Biro, J. Chang, T. Singh, J. Vash, B. Wang, and T. Wang, "A 1.2 tb/s on-chip ring interconnect for 45nm 8-core enterprise xeon processor," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, feb. 2010, pp. 180–181.
- [10] S. Bourduas and Z. Zilic, "A hybrid ring/mesh interconnect for network-on-chip using hierarchical rings for global routing," in *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, may 2007, pp. 195–204.
- [11] F. Sibai, "Adapting the hyper-ring interconnect for many-core processors," in *Parallel and Distributed Processing with Applications, 2008. ISPA '08. International Symposium on*, dec. 2008, pp. 649–654.
- [12] J. C. Villanueva, J. Flich, J. Duato, H. Eberle, N. Gura, and W. Olesinski, "A performance evaluation of 2d-mesh, ring, and crossbar interconnects for chip multi-processors," in *Proceedings of the 2nd International Workshop on Network on Chip Architectures*, ser. NoCArc '09. New York, NY, USA: ACM, 2009, pp. 51–56. [Online]. Available: <http://doi.acm.org/10.1145/1645213.1645226>
- [13] Intel Corporation, *Intel 64 and IA-32 Architectures Optimization Reference Manual*. Intel Corporation, 2011.
- [14] L. Benini and G. D. Micheli, "Networks on chips: A new soc paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, January 2002.
- [15] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-tile 1.28tflops network-on-chip in 65nm cmos," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, Feb. 2007, pp. 98–589.
- [16] Intel, "Single-chip cloud computer," May 2010, <http://techresearch.intel.com/articles/Tera-Scale/1826.htm>.
- [17] T. Lei and S. Kumar, "A two-step genetic algorithm for mapping task graphs to a network on chip architecture," in *DSD, 2003*, sep. 2003, pp. 180–187.
- [18] T. Xu, P. Liljeberg, and H. Tenhunen, "Optimal number and placement of through silicon vias in 3d network-on-chip," in *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2011 IEEE 14th International Symposium on*, april 2011, pp. 105–110.
- [19] D. Abts, N. D. E. Jerger, J. Kim, D. Gibson, and M. H. Lipasti, "Achieving predictable performance through better memory controller placement in many-core cmps," in *Proceedings of the 36th annual international symposium on Computer architecture*, June 2009, pp. 451–461.
- [20] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," in *Proceedings of the 22nd International Symposium on Computer Architecture*, June 1995, pp. 24–36.
- [21] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, October 2008, pp. 72–81.
- [22] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *ACM SIGPLAN*, October 2002, pp. 211–222.
- [23] A. Patel and K. Ghose, "Energy-efficient mesi cache coherence with pro-active snoop filtering for multicore microprocessors," in *Proceeding of the thirteenth international symposium on Low power electronics and design*, August 2008, pp. 247–252.
- [24] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *Computer*, vol. 35, no. 2, pp. 50–58, February 2002.