Exploring Memory Controller Configurations for Many-Core Systems with 3D Stacked DRAMs

Fen Ge[1], Jia Zhan[2], Yuan Xie[2], Vijaykrishnan Narayanan[3]
[1]College of Electronic and Information Engineering, Nanjing Univ. of Aero. and Astro., Nanjing, China
[2] Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, USA
[3]Department of Computer Science and Engineering, Pennsylvania State University, PA, USA
[1]E-mail: gefen@nuaa.edu.cn, [2]E-mail: {jzhan, yuanxie}@ece.ucsb.edu, [3]E-mail: vijay@cse.psu.edu

Abstract

Network-on-Chip (NoC) provides a scalable approach to integrate more and more cores on chip, while limited capacity and bandwidth of DRAMs becomes the performance bottleneck. To break the memory wall, 3D integration of DRAMs and processors using Through Silicon Vias (TSVs) has emerged. Distributed memory controllers (MCs) are allocated on chip in order to utilize the abundant bandwidth of stacked DRAMs, but unavoidably incur significant hardware overhead. In this paper, we analyze the design of memory controllers in NoC-based many-core systems with stack-DRAMs. By analyzing the interaction between NoCs and MCs, the optimal number and placement of MCs are explored. Specifically, a Genetic algorithm (GA) based approach is proposed to find the optimal memory controller placement with different number of DRAM partitions. We evaluate memory controller configurations for various memory-intensive applications in terms of network latency and energy, as well as thermal distribution.

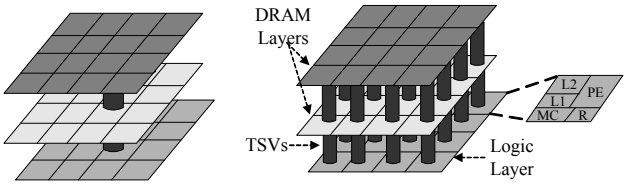
Keywords

Network-on-Chip, memory controller, 3D Integration

1. Introduction

Network-on-Chip (NoC) provides an efficient and scalable interconnect for many cores and large caches in chip multiprocessors (CMP). However, due to the limited number of pins in the processor's package, the available capacity and bandwidth of DRAMs becomes the performance bottleneck, also known as memory wall. In order to break the memory wall, three-dimensional (3D) integration of DRAMs and processors has received enormous attention recently [1,2,3]. 3D integration enables using dense Through Silicon Vias (TSVs) as the vertical interconnects to stack DRAMs directly on top of processors.

Previous studies have already started exploring the performance benefits of 3D integrated architecture with stacked DRAMs. The performance impact of placing the main memory system on top of the processor using 3D fabrication technologies was studied [4,5]. Several stacked DRAM architectures for CMPs were proposed [1,2,4,6,7,8, 9]. The high-level view of the prior proposed architectures



(a) Centralized architecture (b) Distributed architecture

Figure 1: The prior proposed 3D stacked DRAM architectures.

is shown in Fig.1. Fig.1a represents a **centralized** 3D stacked DRAM architecture in which the design of the DRAM layer is almost unchanged from the traditional off-chip DRAM. In this architecture, the DRAM is connected to the processors in the logic layer via the central memory interface. This architecture has the smallest area but is not adequate for providing high bandwidth and large memory level parallelism (MLP). Fig. 1b shows a generic **distributed** 3D stacked DRAM architecture. In this architecture, the memory interface is composed of multiple DRAM memory controllers (MCs). On the bottom logic layer, processing elements (PEs) are connected through NoC. Each PE can directly access the DRAM module that is stacked on top of it by its local MC and the vertical TSV bus. PEs can also access DRAMs on top of other PEs by transferring requests and data through horizontal NoC routers and links. This architecture can improve the performance of the memory accesses with large MLP. However, equipping an MC for each PE can lead to high resource overhead, since each MC consists of a read queue, a write queue and a command queue. On the other hand, in this generic distributed architecture, each MC is associated with the same number of TSVs providing control signals and the data bus to the stacked DRAMs. Massive TSVs provide abundant bandwidth but the implementation of TSVs requires extra manufacturing cost and silicon area, which can possibly reduce the chip yield [10]. Therefore, as the number of PEs grows, it is not practical and necessary to assume each PE will have an MC directly attached. To find a balance between performance and number of MCs and associated TSVs is becoming critical. Furthermore, a many-core system with *n* PEs and *m* MCs raises another question of *where* the MCs should be placed in the logic layer interconnected through NoC. Thus, the configurations of MCs with different number and location should be explored in the stacked DRAMs architecture.

This work was supported by the Natural Science Foundation of China (No. 61106018, 61376025) and NSF 1500848, 1017277.

There have been several work for exploring design space of distributing memory controllers. Abts et al. [11] first investigate MC configurations in many-core CMP. This work explores placements of a fixed number of MCs which are used to access off-chip DRAMs. However, the proposed MC placements do not apply to the stacked on-chip DRAM architecture. Chen et al. [12] proposes a distributed memory interface synthesis framework for application-specific NoCs with stacked DRAMs. The goal of the framework is to minimize the number of TSVs while the user-defined performance constraint is met. The configurations with different number of MCs and TSV width are analyzed, but the placements of MCs are not discussed in their work.

The strategy of MC placement for off-chip DRAMs and 3D stacked DRAMs is different. For off-chip DRAMs, MCs are usually placed on the edge due to the requirement of placed pins, while for 3D stacked DRAMs, the placement of MCs is more flexible. But stacked DRAMs also induce some access constraints via vertical TSVs.

In this paper, we are targeting a general purpose design rather than an application specific customized design, and the paper makes the following major contributions:

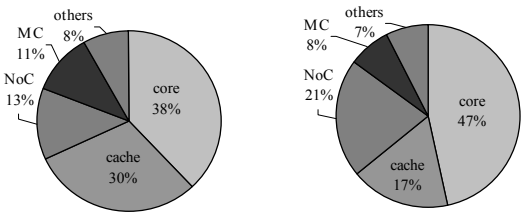
- Conducts a comprehensive design space exploration on the number and placement strategies of MCs, and analyzes their impact on the performance and cost of NoC-based many-core systems.
- Proposes a Genetic algorithm (GA) based approach to obtain optimal MC placements with different DRAM partitions stacked on top of the PEs.
- Evaluates memory controller configurations with various number and optimal placements for memory-intensive applications in terms of network latency, energy, and thermal distribution.

2. Motivation Examples

Throughout the paper, we target a NoC based many-core system with 3D stacked DRAM as shown in Fig.1. The architecture contains two layers. The bottom layer is partitioned into regular tiles connected by a 2D mesh network. Each tile has a PE with or without a MC, a private L1 instruction/data cache, a bank of shared L2 cache, and a router. The top DRAM layer is partitioned into multiple modules. Each MC is in charge of only one DRAM module stacked on top of it via vertical TSVs.

In such architecture, it remains a challenge to determine the optimal number of MCs and how these MCs should be placed, in order to provide the best network performance for accessing different DRAM modules with least cost.

The cost mainly contains area overhead and power consumption on MCs. For illustration, the area and power breakdown of different components when scaling the number of cores to 16 based on the Niagara2 [13] processor is shown in Fig.2. We evaluate the area overhead and power dissipation with McPAT [14] for 16 cores, 16 banks of L2 caches, 4 MCs, NoC, and others (PCIe controllers, etc.) at the 32nm technology node. We can observe that the MC cost contributes a non-negligible portion to the total chip cost, especially compared with the NoC cost.

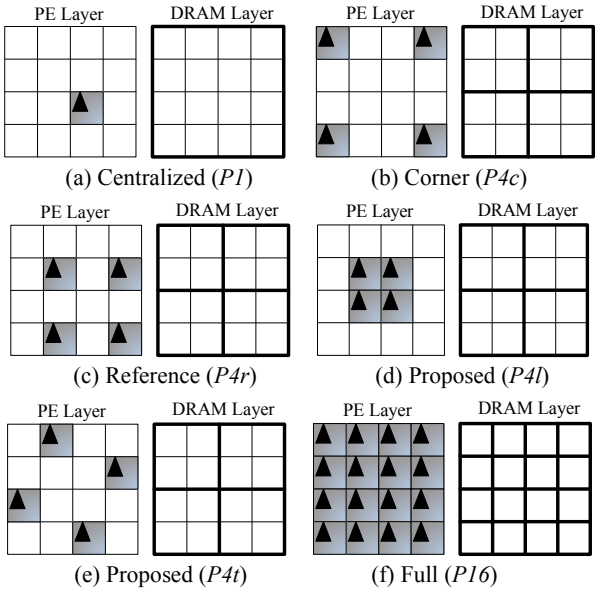


(a) Area breakdown (b) Power breakdown

Figure 2: Area and power breakdown of a CMP.

Besides, each MC is associated with TSVs to access the top DRAM. The TSVs also occupy a considerable portion of die area and can possibly reduce the die yield [10]. As the number of TSVs grows, the probability of a faulty chip die becomes higher. Assuming that the channel width in a NoC is 128 bits, full MC configuration for an 8×8 NoC would require 128×8×8=8192 TSVs for parallel data signals. According to a recent statement [10], the yield is only 36.79% for 10,000 TSVs. Therefore, *how many* MCs should be placed in the many-core system is worth investigating.

Fig.3 illustrates some MC configurations in a 4×4 NoC with stacked DRAMs. The centralized placement with only one MC and the generic distributed placement with full MCs are shown in Fig. 3a and Fig.3f, respectively. Fig. 3b presents the placement adopted in Tilera's TILE64 chip [15] with four MCs on the corner. Fig. 3c describes the placement with four MCs presented by Chen et al [12]. Two placements by our proposed approach are shown in Fig.3d and Fig. 3e. The former is a latency-aware placement, and the other is a thermal-aware design (details will be discussed in Section 3 and 4). These placements are presented by *Pm(c,r,l,t)*, where *m* denotes the number of MCs, *c, r, l, t* denote the corner, reference, latency-aware, and thermal-aware placement with the fixed MC number, respectively.

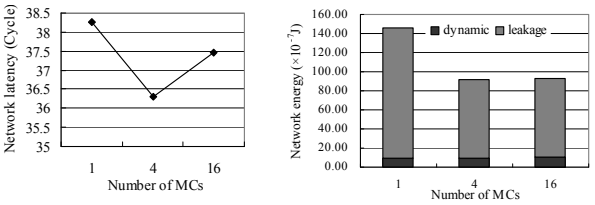


(a) Centralized (*P1*) (b) Corner (*P4c*)

(c) Reference (*P4r*) (d) Proposed (*P4l*)

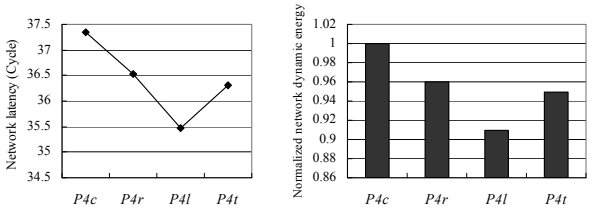
(e) Proposed (*P4t*) (f) Full (*P16*)

Figure 3: Some MC configurations with DRAM partitions.



(a) Latency variation (b) Energy variation

Figure 4: Network latency and energy variation with different number of MCs.



(a) Latency variation (b) Dynamic energy variation

Figure 5: Network latency and dynamic energy variation with different placements.

Fig. 4 shows the network average latency and energy variation with different number of MCs for NoC when executing *disparity*, an application from the San Diego Vision Benchmark Suite [16]. Fig. 5 shows the network average latency and dynamic energy variation with different placements under the fixed four MCs. We can observe that, four MCs with four DRAM partitions achieve the optimal balance between performance and cost, in comparison with centralized and full placements. Among the placement options with four MCs, our proposed results outperform the other two (details will be discussed in Section 4). Therefore, *where* these MCs should be placed is also critical.

3. The proposed placement approach

As discussed in the previous section, it is necessary to find an optimal placement of MCs with different DRAM partitions in the stacked DRAM architecture, since the different placement options have an impact on the network latency and energy. In this section, we first present the problem formulation of this configuration issue. Then we describe our proposed placement approach.

3.1. Problem Formulation

In the target architecture, the bottom NoC composed of n regular tiles is modeled by an architecture graph $N(T,L)$, where $T=\{t_1,\ldots, t_n\}$ is the set of tiles and L is the set of links between tiles.

The top DRAMs are also arranged in regular tiles [2], where the number of the DRAM tiles is the same as the number of NoC tiles. Let $R=\{r_1,\ldots, r_n\}$ be the set of tiles in the stacked DRAMs. Each DRAM tile r_i is located on top of the NoC tile t_i. In the system design, several DRAM tiles can be clustered into an accessible DRAM module. We use m non-empty clusters $RC_1,\ldots,RC_m$ to denote the DRAM modules. We also assume the size of each DRAM module is the same, thus each cluster RC_i (i = 1, 2 , ..., m, $m\leq n$) contains k DRAM tiles, where $k=n/m$.

The MC configuration is specified by the number and location of MCs allocated in NoC. In this paper, as shown in Fig.3, we assume an MC only controls a DRAM module stacked on top of it. We use $PMC=\{pmc_1,\ldots, pmc_x\}$ to denote the set of eligible MC placements. Each pmc_i in PMC is a binary array of n elements. If the j-th binary element in pmc_i is 1, an MC is allocated in tile t_j when pmc_i is selected. We use $tpmc_i(RC_j)$ to denote the tile that contains the MC, which controls the DRAM module RC_j.

For a selected MC configuration, access latency Lat of transferring a packet between PEs and DRAM modules is chosen as a metric to evaluate the performance of the configuration. Lat can be given by

$$Lat = Lat_{NoC} + L_{m-access} \quad (1)$$

where Lat_{NoC} and $L_{m\text{-}access}$ denote on-chip transfer latency and memory access latency, respectively. Lat_{NoC} can be further modeled by

$$Lat_{NoC} = \sum_{j=1}^{m} \sum_{\forall t_i \in T} H(t_i, RC_j) \quad (2)$$

where $H(t_i, RC_j)$ denotes the Manhattan distance between the PE in tile t_i and the tile $tpmc_i(RC_j)$. $L_{m\text{-}access}$ includes queuing delay Q_{mc} in MC and access latency λ_{DRAM} of a DRAM module. λ_{DRAM} is assumed to be same for each MC accessing the top DRAM module, as well as Q_{mc} is assumed to be identical in our design.

The goal of our work is to find MC configurations that provide good performance under limited number of MCs. From (1) and (2), we can observe that $H(t_i, RC_j)$ should be minimized in order to reduce access latency. Also, by reducing the variance in access latency, the configuration can provide fair access to each MC and make the on-chip network less sensitive to the PE on which an application is executed. Furthermore, MC placements should be more distributed without causing local congestion and hotspots.

Based on the above analysis, the MC configuration problem for many-core system with stacked DRAMs can be formulated as follows.

Given an NoC architecture $N(T,L)$ and the partitioned DRAM module clusters $RC_1,\ldots,RC_m$;

Find the optimal MC placement pmc, **such that**

- the average access latency from each PE to all the allocated MCs is minimized, i.e.,

$$\min Avg(H(t_i, RC_j)) = \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n} H(t_i, RC_j)$$

- the variation in access latency from all PEs to each MC is minimized, i.e.,

$$\min Sd(H(t_i, RC_j))$$

$$= \sqrt{1/m\sum_{j=1}^{m}(\sum_{i=1}^{n} H(t_i, RC_j) - Avgmc(H(t_i, RC_j)))^2}$$

$$\text{where } Avgmc(H(t_i, RC_j)) = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} H(t_i, RC_j)$$

- the allocation among any two MCs is distributed with minimal variation, i.e.,

$$\min Distr(H(RC_i, RC_j)) = \frac{Sd(H(RC_i, RC_j))}{Avg(H(RC_i, RC_j))}$$

where $H(RC_i, RC_j)$ denotes the Manhattan distance between the tile *tpmc* (RC_i) and the tile *tpmc* (RC_j).

3.2. Placement Approach

An on-chip network with n tiles and m memory controllers will have C_n^m possible permutations that must be compared against each other in order to choose the best. To deal with this complexity we use the Generic Algorithm (GA) which takes heuristic-based approach to optimization. Our GA based placement approach consists of four steps described as follow.

● *Initialize population*. A MC placement *pmc* is represented by a chromosome, and an amount of chromosomes compose an initial population. Each gene in the chromosome contains a binary bit generated randomly, as shown in Fig. 6. To ensure the rule that a DRAM module is only controlled by one MC located on the bottom region, the number of "1" in a chromosome must equal to the number of the partitioned DRAM module clusters.

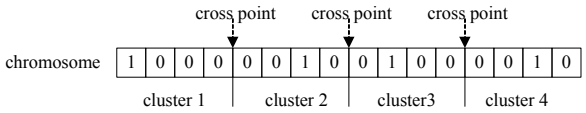


Figure 6: The proposed GA encoding and crossover strategies.

● *Evaluate fitness*. Each chromosome in the population is evaluated for its fitness. In our work, the fitness function is given by the reciprocal of the following cost function, which takes the above three requirements into consideration.

$$cost = \alpha \times Avg(H(t_i, RC_j)) + \beta \times Sd(H(t_i, RC_j)) + \gamma \times Distr(H(RC_i, RC_j)) \times \varepsilon \quad (3)$$

where α, β, and γ are the weighting parameters (0<α, β, γ<1, and α+β+γ=1). ε is a compensation factor to ensure the three requirements in the same scale.

● *Create new population*. Three operators (selection, crossover and mutation) are applied to create a new population. The selection operation selects two parent chromosomes from the population with probability proportional to the potential parents' fitness. The crossover operation randomly selects cross points from the parents to form a new generation. For mutation, we exchange two genes in the new chromosome that are randomly selected. We should pay attention to the fact that the crossover and mutation operations may create illegal answers for our MC allocation problem. At this stage, there are possibilities that more than one MC may be allocated in the same cluster region, or no MC is allocated. Therefore, the cross points are limited in the edge of the cluster in the chromosome, as shown in Fig. 6. Also, additional validity check should be applied after mutation.

● *Output optimal result.* The approach repeats the above two steps until no improvement in observed fitness over for a sufficient number of iterations, then reports best result as the generated optimal MC placement in the NoC.

3.3. Placement Results

Given the NoC architectures with different topology sizes and different number of the partitioned DRAM module clusters, the generated optimal MC placements are shown in Table 1.

Table 1: Optimal MC placements for different architectures and optimization goals

size	# of DRAM clusters	α=β=0.4, γ=0.2	α=β=0.25, γ=0.5
4×4	2	5, 10	5, 10
	4	5, 6, 9, 10	1, 7, 8, 14
	8	1, 2, 5, 6, 9, 10, 13, 14	1, 2, 4, 7, 8, 11, 13, 14
8×8	2	27, 36	27, 36
	4	20, 26, 37, 43	1, 14, 45, 49
	8	11, 12, 20, 26, 37, 42, 51, 53	10, 12, 21, 25, 38, 42, 52, 58
	16	3, 6, 9, 12, 18, 22, 25, 29, 41, 43, 44, 46, 49, 51, 52, 54	9, 11, 13, 14, 18, 22, 24, 29, 39, 41, 43, 44, 51, 52, 54, 57

According to the cost function given by (3), the value of the weighting parameters influences the placement results. If α, β>γ, the optimization goal is access latency rather than MC distribution. On the contrary, if γ>α, β, the optimization goal is MC distribution rather than access latency. Also, we limit the difference between α, β and γ, in order to keep the balance. In our case for latency-aware design, we set α=β=0.4, and γ=0.2; for thermal-aware design, we set α=β=0.25, and γ=0.5. The placement results with different optimization goals are shown in Table 1, which are represented by the tile index on the bottom layer from the top-left corner to the bottom-right corner.

4. Experimental results

In order to evaluate the proposed MC configurations, we conduct some real trace simulations in this section.

4.1. Platform Setup

We use a functional simulator based on Pin [17] to collect memory access traces from applications, which are then fed into a cycle-level trace-driven multicore simulator integrated with Garnet [18] and DSENT [19] to evaluate the network performance and power. The detailed configurations of the NoC with stacked DRAMs platform used for experiments are listed in Table 2.

Table 2: System and interconnect configurations

core count	16
L1 I & D cache	private, 16KB per tile
L2 cache	shared, 128KB per tile
cacheline size	64B
frequency	1GHz
topology	4×4 mesh
routing	XY routing
VC count	4 VCs per port
buffer depth	4 buffers per VC
packet length	5 flits, 8B/flit

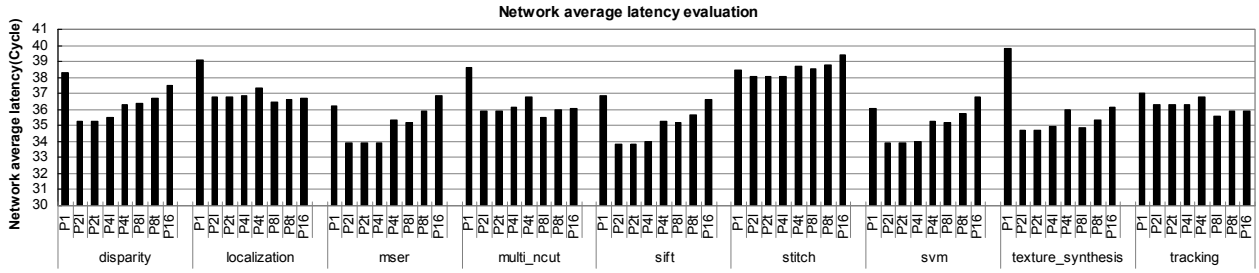


Figure 7: The network average latency for different MC configurations.

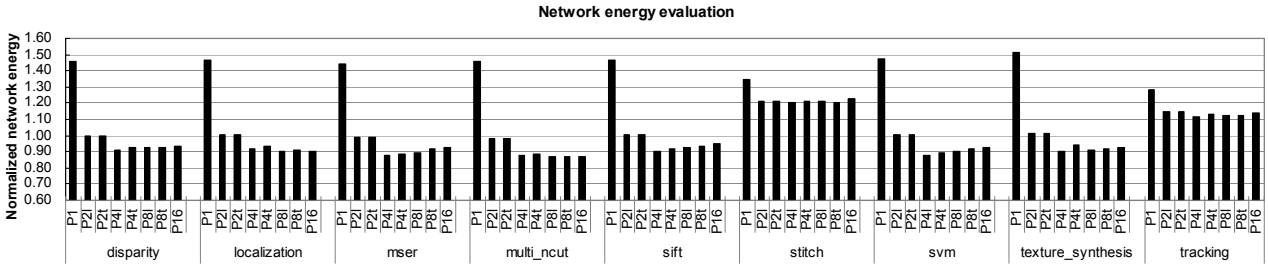


Figure 8: The network energy evaluation for different MC configurations.

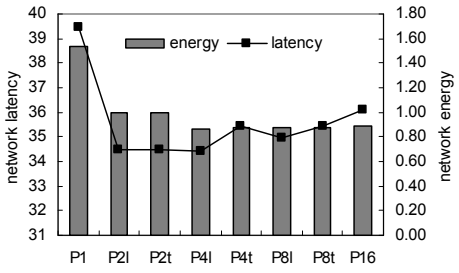
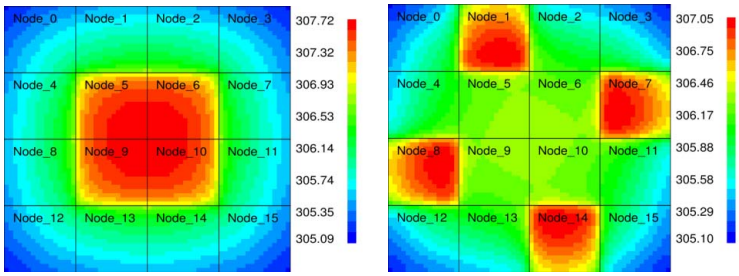


Figure 9: Evaluation for random application mapping.



(a) Latency-aware placement (b) Thermal-aware placement

Figure 10: The temperature distribution for different placements

4.2. Performance and Energy Evaluation

We use a diverse set of memory-intensive applications from the San Diego Vision Benchmark Suite [16] for network performance and power analysis. We run different benchmarks with different MC configurations as presented in Table 1.

Fig. 7 shows the network average latency for different applications. The figure presents the same trend as that shown in the motivation example (Fig. 4a). That is, compared to the centralized configuration with only one MC, the network average access latency is reduced as the number of the allocated MCs increases. However, the general distributed configuration with the maximum number of the allocated MCs is not the best. In the MC configuration problem, with the number of MCs increasing, the number of DRAM partitions is increasing, while the memory capacity controlled by one MC is decreasing. In this situation, PEs in one cluster have to transfer requests and data through NoC routers and links to the MCs allocated in other clusters, which lead to additional network latency. The optimal number of allocated MCs may vary for different applications. For instance, the choice of two MCs is better for *disparity* and *sift*, while four MCs configuration is better for *mser* and *svm*.

Fig.8 shows the network energy evaluation for different applications. As presented in section 2, the MC energy consumes a non-negligible portion to the total chip energy, and obviously it should increase as the MC number increases. Therefore, we only need to investigate the impact of MC configurations on network energy. As the result shown in Fig. 4b, the network leakage energy consumes a significant portion of total network energy. The network with fewer number of MCs may cause more congestion for memory accesses and need to spend more cycles on completing the transfer, which lead to larger leakage energy consumption. Therefore, the total network energy consumption decreases as the number of MCs increases (the values shown in the Fig. 8 are normalized to the energy consumption caused by the MC configuration *P2l*).

Besides the impact of the MC number discussed above, we also evaluate the impact of the MC placement on network latency and energy. In the motivation section, we present an example of different placements with four MCs (Fig. 5). It shows that the placements generated by our proposed approach achieve lower cost in network latency

and energy compared with the common design and reference design. In this section, we compare two kinds of placements generated by our GA based approach with different optimal goals (*Pml* refers to latency-aware placement, and *Pmt* refers to thermal-aware placement, *m*=2,4,8). It can be observed from Fig.7 and Fig.8 that, the placements *Pml* outperform the placements *Pmt* due to the fact that the minimization of the average access distance is mainly taken into consideration in the placements *Pml*.

Furthermore, we also consider the impact of application mapping on the performance and energy. The applications {*disparity, localization, mser, multi_ncut, sift, svm, texture_synthesis*} are randomly mapped to the 16 cores. Fig. 9 shows the results (the energy values shown in the Fig. 9 are normalized to the energy consumption caused by the MC configuration *P2l*). It can be observed that the placements *Pml* achieve better results than the placements *Pmt*, regardless of the core on which an application is executed.

4.3. Thermal Analysis

In the NoC, the node with MC obviously consumes more power than the node without MC. Therefore, the placement of MCs affects the power density of the chip. In our proposed placement approach, we try to scatter the MCs in order to provide better thermal distribution, and thus eliminating thermal hot spots.

In the experiment, we collect the NoC router and MC power densities using McPAT and feed them into a thermal modeling tool HotSpot [20]. The stable temperature of network nodes with different MC placements is shown in Fig.10. As shown in Fig. 10a, four MCs are allocated in the center of the network in order to obtain better access latency, but it results in an overheated spot in the center. In contrast, the thermal-aware placement distributes the MCs over the network, and generates better temperature profile as shown in Fig. 10b.

5. Conclusion

In this paper, we target a NoC-based many-core system with stacked DRAMs, and explore how the number and location of the memory controllers affect the network performance and energy. A GA based approach is proposed to find the optimal location for different number of DRAM partitions, in order to minimize end-to-end on-chip memory access latency and obtain better thermal distribution. Based on the results generated by the proposed approach, various MC configurations are evaluated for memory-intensive applications.

6. References

[1] G. H. Loh, "3D-Stacked Memory Architectures for Multi-core Processors," in *ISCA*, 2008, pp.453-464.

[2] D. Woo, N. Seong, D. Lewis, and H. Lee, "An optimized 3d-stacked memory architecture by exploiting excessive, high-density tsv bandwidth," in *HPCA*, 2010, pp. 1-12.

[3] T. Zhang et al., "3D-SWIFT: A High-performance 3D-stacked Wide IO DRAM," in *GLSVLSI*, 2014, pp. 51-56.

[4] C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari, "Bridging the processor-memory performance gap with 3d ic technology," *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 556-564, 2005.

[5] G. L. Loi et al., "A thermal-aware performance analysis of vertically integrated (3-d) processor-memory hierarchy," in *DAC*, 2006, pp.991-996.

[6] T. Kgil et al., "Picoserver: using 3d stacking technology to enable a compact energy efficient chip multiprocessor," in *ASPLOS*, 2006, pp. 117-128.

[7] A. Marongiu, M. Ruggiero, and L. Benini, "Efficient openmp data mapping for multicore platforms with vertically stacked memory," in *DATE*, 2010, pp. 105-110.

[8] I. Loi et al., "An efficient distributed memory interface for many- core platform with 3d stacked dram," in *DATE*, 2010, pp. 99-104.

[9] M. Jun, M. Kim, and E. Chung, "Asymmetric dram synthesis for heterogeneous chip multiprocessors in 3D-stacked architecture," in *ICCAD*, 2012, pp. 73-80.

[10] T. C. Xu et al., "Optimal placement of vertical connections in 3D network-on-chip," *Journal of System Architecture*, vol. 59, no. 7, pp.441-454, 2013.

[11] D. Abts et al., "Achieving predictable performance through better memory controller placement in many-core CMPs," in *ISCA*, 2009.

[12] Y. Chen, C, Yang, and J, Chen, "Distributed memory interface synthesis for network-on-chips with 3d-stacked drams," in *ICCAD*, 2012, pp.458-465.

[13] U. G. Nawathe et al., "Implementation of an 8-core, 64-thread, power-efficient sparc server on a chip," *IEEE JSSC*, vol.43, no.1, pp. 6–20, 2008.

[14] S. Li et al., "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.

[15] D. Wentzlaff et al., "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15-31, 2007.

[16] S. K. Venkata et al, "Sd-vbs: The san diego vision benchmark suite," in *IISWC*, 2009, pp. 55-64.

[17] H. Patil et al.,"Pinpointing representative portions of large intel itanium programs with dynamic instrumentation," in *MICRO*, 2004, pp.81-92.

[18] N. Agarwal, et al, "GARNET: A detailed on-chip network model inside a full-system simulator," in *ISPASS*, 2009, pp. 33-42.

[19] C. Sun et al., "DSENT-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *NOCS*, 2012, pp. 201-210.

[20] W. Huang et al., "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. on VLSI*, vol. 14, no. 5, pp. 501–513, 2006.