Thermal-aware Energy Optimization by Synthesizing Firm 3-D Network-on-Chip with Voltage-Frequency Islands

Song Jin*, Jun Liu#,Yu Wang*

*Department of Electronic and Communication Engineering, School of Electrical and Electronic Engineering, North China Electric Power University, P. R. China

#School of Computer and Information, Hefei University of Technology, Hefei, P. R. China

jinsonng@163.com

***Abstract*—In this paper, we introduce voltage-frequency island (VFI) -based design paradigm into three dimensional (3-D) network-on-chip (NoC) to optimize system energy. The prominent challenges to VFI-based 3-D NoC designs are the exacerbated thermal issues. Moreover, targeting hard platform with pre-designed structure restrains the optimization space of the prior work. In view of the above limitations, we propose a VFI-aware synthesis framework to minimize system energy and keep thermal balancing for the firm 3-D NoC platform where designers have the freedom to make mapping decisions on computation components. Besides task scheduling and voltage scaling for computation energy minimization, core stacking and task migrating algorithm are proposed to optimize communication energy and balance powers across the core stacks. By treating each core stack as a unity, VFI aware 3-D NoC mapping problem can be simplified as the mapping issue on 2-D. Experimental results demonstrate that on average our framework can achieve an energy reduction of 18.6% over the prior thermal balancing algorithm. Moreover, on average 5.7 ℃ reduction in peak temperature is achieved by our framework, compared with the state-of-art energy optimization scheme.**

Keywords- System energy, Voltage-frequency island, 3-dimensional, System-on-chip

I. INTRODUCTION

With abundant energy efficient interconnects, 3-D network-on-chip (NoC) brings great potential to build a complex system with lower system energy [1, 2]. Moreover, the tile-based NoC structure fits well with the concept of voltage-frequency island (VFI) [3]-[5], a kind of globally asynchronous locally synchronous (GALS) design paradigm. With VFI design, the tiles in NoC are divided into different islands, and each island can operate at its own voltage and frequency. Such design style helps to implement fine-grained system-level power management, thus providing a vigorous way to build an energy efficient 3-D system.

However, the rigorous thermal issues bring significant challenges on introducing VFI-based design into the 3-D NoC platform. The increased power density in 3-D stacking exacerbates hot spot and generates high temperature on the chip [6]. Heterogeneous workloads executed on 3-D NoC causes power variation, resulting in the thermal gradient across the chip. High temperature and thermal gradient not only degrade system performance and reliability, but also offset the effort of system energy optimization.

Although VFI-based energy optimizations have been popularized in 2-D NoC [7]-[10], the more complex thermal characteristics of 3-D integration prevent them from being effectively applied to the 3-D platform. On the other hand, the existing 3-D thermal optimization solutions, either in hardware [11]-[13] or in software level [14]-[16], generally did not take VFI-based design into consideration and ignored energy optimization. Moreover, the prior work commonly targeted hard NoC platform where both computation and communication components have been fixed. As a result, optimization space is inevitably restrained.

To tackle above problems, we propose to synthesize 3-D NoC with VFI for minimizing system energy meanwhile keeping thermal balancing across the chip. Unlike prior work, our synthesis framework targeted firm NoC platform where the communication architecture has been pre-designed but the mapping of the computation components has not been determined yet [17]. This permits us to unified consider NoC mapping and VFI partitioning and enables us to flexibly explore better thermal-energy optimization tradeoff. Besides exploiting task scheduling and voltage scaling for computation energy minimization, core stacking and task migrating algorithm is proposed to optimize communication energy and balance powers across the core stacks. By treating each core stack as a unity, VFI aware 3-D NoC mapping problem can be simplified as the mapping issue on 2-D. Experimental results demonstrate that on average our framework can achieve an energy reduction of 18.6% over the thermal balancing algorithm. While comparing with the state-of-art energy optimization schemes, on average 5.7 ℃ reduction in peak temperature is achieved by our framework.

The rest of the paper is organized as follows. Section Ⅱ introduces related work. Section Ⅲ presents the preliminaries and framework overview. Section Ⅳ details the proposed synthesis framework. Section Ⅴ presents the experimental results. We conclude in Section Ⅵ.

II. RELATED WORK

The large amount of literature has been proposed to handle thermal issues in 3-D chip. In hardware aspects, Goplen and Wong separately proposed thermal vias insertion [11] and placement [12] to reduce the chip temperature. Bakir et al. [13] proposed to adopt liquid cooling to help heat dissipation for 3-D ICs. In software aspects, dynamic thermal management (DPM) technologies were proposed to restrain thermal emergency at runtime [14]. Zhou et al. [15]

978-1-4799-3435-5/14 $31.00 © 2014 IEEE
DOI 10.1109/ICMTMA.2014.193

proposed thread allocation and scheduling algorithm for thermal balancing on 3-D multi-core platform. Zhu et al. [16] proposed to combine task scheduling and voltage scaling for 3-D MPSoC thermal optimization. Above work, however, conducts at the fixed hardware platform and mainly focuses on thermal reduction while ignoring energy optimization.

As for energy optimization on VFI-based design, Ogras et al. [7] proposed VFI partition and voltage/frequency assignment to minimize the energy in 2-D NoCs. References [8] and [9] proposed to combine voltage scaling with the task scheduling algorithm to minimize computation and communication energies in 2-D multi-core. Jiang et al. [10] proposed a VFI-aware energy optimization framework for 2-D NoCs, which maps cores and determines routing at design time. Due to the large difference of thermal features between the 3-D chip and 2-D counterpart, above work cannot be applied directly to the 3-D platform. Targeting 3-D multi-core chip, Cheng et al. [18] proposed a thermal-constrained task allocation algorithm to minimize communication energy. However, the algorithm ignores the computation energy.

III. PRELIMINARIES

Figure 1 (a) sketches a homogeneous multi-core 3-D NoC platform with VFI design, which is tile based NoC-bus structure [20]. Communication within a layer is implemented by the mesh NoC while cross layer communication is achieved by multi-drop TSV bus. Each tile contains a core and a router. The mixed-voltage/mixed-frequency FIFOs located on the VFI boundaries are used for data synchronization. As for the target application, we focus on high deterministic applications described as the communication task graphs

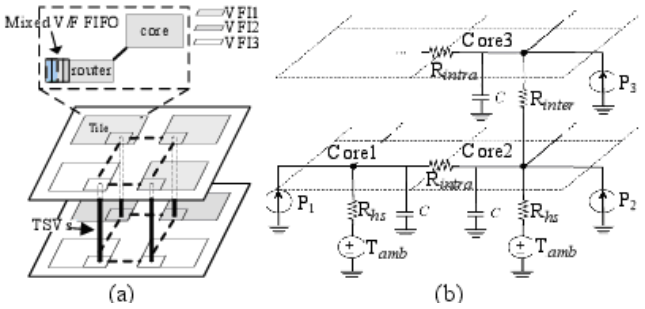

Figure 1. (a) VFI-based 3D NoC, and (b) thermal model (from [19]).

A. Thermal Model for 3-D NoC

Figure 1 (b) illustrates the thermal model for NoC based 3-D multi-core. Given the thermal model, temperatures of core 3 nd core 2 can be expressed as:

$$\begin{aligned} T_3 &= P_3 \cdot R_{inter} + T_2 \\ T_2 &= (P_2 + P_3) \cdot R_{hs} + T_{amb} \end{aligned} \quad (1)$$

where P_3 and P_2 denote the powers consumed by cores 3 and 2. R_{inter} denotes the thermal resistance between cores 3 and 2 in the vertical direction. R_{hs} denotes the thermal resistance from core 2 to ambient. T_{amb} is the ambient temperature.

From Eq.1 we know that with the constant thermal resistance parameters, temperature of the core is mainly decided by the power that the core consumed. Moreover, stronger thermal correlation exists between vertically stacked cores ($R_{intra} \approx 16 R_{inter}$) [19]. As a result, to keep thermal balancing across the 3-D NoC chip, it is essential to keep power balancing across the core stacks [15].

B. Energy Model for VFI-based 3-D NoC

Based on above VFI-based 3-D NoC platform, the total system energy can be divided into computation energy Ecomp and communication energy Ecomm , respectively. The total computation energy comes from the execution of the task graph and can be expressed as:

$$E_{comp} = \sum_{i=1}^{n} \left(NC_j \cdot C_i \cdot V_i^2 \right) \quad (2)$$

where NC_j denotes the execution cycles of task j. C_i stands for the total switching capacitance per cycle of core i. V_i denotes the voltage that the core i operated on. n is the total number of the cores on the chip.

For calculating the total communication energy, as in [7] and [18], the bit energy consumed on transmitting one bit from core i to core j is defined as:

$$E_{bit} = \sum_{i}^{n_V} \left(E_{bit}^{R}(i) + E_{bit}^{Link}(i) + E_{bit}^{FIFO}(i) \right) \cdot \frac{V_i^2}{V_{DD}^2} \quad (3)$$

where E_{bit}^{R}, E_{bit}^{Link} and E_{bit}^{FIFO} are bit energies consumed on the router, the link and the mixed-voltage/mixed-frequency FIFOs, respectively. n_V is the number of VFIs. V_i is the supply voltage that VFI_i operated on. Based on E_{bit}, the total communication energy consumed by a task graph with m communication transactions can be formulated as:

$$E_{comm} = \sum_{k=1}^{m} E_{bit} \cdot Q_k \quad (4)$$

where Q_k denotes the data volume in communication transaction k between core i and core j.

IV. DETAILS IN SYNTHESIS FRAMEWORK

Step 1: energy aware task scheduling and V/F scaling. It is essential to unified consider the task scheduling and the subsequent V/F scaling to minimize the total computation energy meanwhile still meeting the deadline constraints. Our task scheduling algorithm hence allocates more slack to the energy hungry task and tries to schedule the tasks with close slacks onto the same core. With more slack allocated, operating voltage of the energy hungry task can be reduced as low as possible, thus achieving maximum energy reduction. Moreover, scheduling the tasks with close slacks onto the same core can avoid one or a few of scheduled tasks with small slacks restrict voltage scaling of the core, which also helps to reduce the energy. The pseudo-code of our algorithm is as shown in Figure 2. Note that core stacking and NoC mapping have not been determined at this time.

Algorithm of energy aware task scheduling

Input: task graph, core set; Output: task schedule;

1. Calculate E_{comp} for each task at nominal V/F;
2. Calculate total slack and E_{comp} for each path in task graph;
3. **for** each task on a path
4. task.*slack*=(task.E_{comp}/path.*total*E_{comp})×path.*totalslack*;
5. **while**(Full Task List is not empty){ //Start task scheduling
6. For each ready task, identify the available core on which the scheduled tasks have close slacks with it;
7. Schedule ready tasks onto available cores;
8. Remove the scheduled ready tasks from Full Task List;}

Figure 2. Energy aware task scheduling algorithm.

After the task scheduling, the lowest operating voltages of the cores are then determined for computation energy minimization. Meanwhile, the scaled voltages of the cores are also as the indication for VFI partition in the last step.

Step 2: core stacking and task migrating. Above task scheduling and V/F scaling can achieve minimum computation energy. However, the communication energy does not necessarily be minimized. Moreover, the variation on the powers of scheduled tasks across individual cores still remains. In this step, we perform core stacking to reduce the communication energy and exploit task migrating to balance the powers across the core stacks. In core stacking, the cores with maximum communication quantity are grouped into one stack. This can reduce communication energy significantly since bit energy in vertical link is far more less than the one in horizontal link. Here communication quantity between two cores is actually the sum of communication quantities of the scheduled tasks on these two cores. Next, task migrating is performed to migrate or swap the scheduled tasks among the cores for balancing powers across the core stacks. Note that we try to balance the powers across the core stacks for all the time segments during the total execution time of the task graph. The division of time segments is implemented in a coarse grained manner by aligning execution times of the scheduled tasks among all the cores. Below a simple example is used as an illustration.

A task graph with total 16 tasks is scheduled onto 4 homogeneous cores by applying the task scheduling algorithm and the V/F scaling process in step 1. Then, the total execution time of the task graph is divided into 7 time segments (t1 to t7) based on the execution times of the scheduled tasks. The scheduling results are as shown in Figure 3(a). As for core stacking, we found that cores 1, 4 and cores 2, 3 have largest communication quantities. Hence we group cores 1, 4 and cores 2, 3 into one stack, respectively.

Above core stacking does not necessarily help to balance the powers across the stacks. As shown in Figure 3(b), to balance the stack powers in t3, task 3 can be migrated from core 1 to core 2 because the idle interval between execution times of tasks 1 and 6 is enough to accommodate the execution of task 3. Similarly, task 10 can be migrated from core 2 to core 4 for balancing the stack powers in t6. As for time segment t4, task 7 cannot be migrated from core 4 to core 2 originally due to the idle interval between execution times of tasks 2 and 9 is less than the execution time of task 7. Fortunately, task 2, which is also the precedent task of task 7, remains some slack after V/F scaling. We can squeeze the slack of task 2 and make execution time of task 7 to move ahead, which allows the migration of task 7 from core 4 to core 2. The overall core stacking and task migrating algorithm is as shown in Figure 4.

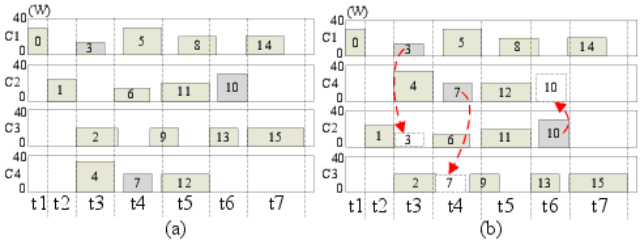

Figure 3. Task migrating illustration, (a) original task schedule and (b) after task migrating.

Algorithm of core stacking and task migrating

Input: original task schedule; core set;
Output: new task schedule; core stacks;
//Core stacking

1. Calculate communication quantity Q_{pairs} for each core pairs;
2. Group two cores with largest Q_{pairs} into one stack;

//Task migrating

3. **for** each time segment {
4. Calculate sum of the powers for each core stack;
5. Identify the tasks to be migrated or swapped;
6. Check the idle interval or remaining slack of the precedent task
7. **if** (idle interval or remaining slack is allowed){
8. Task migrating or swapping;}}

Figure 4. Core stacking and task migrating algorithm.

Step 3: VFI aware 3-D NoC mapping. After the core stacking, we perform NoC mapping in this step to determine the final hardware configurations. During the NoC mapping process, VFI partition is taken into consideration since operating voltages of the cores have been determined in step 1. Moreover, each core stack is treated as a unity to be mapped into the tiles in NoC, which enables us to transform NoC mapping problem in 3-D into a mapping issue in 2-D. A heuristic algorithm for 2-D NoC mapping proposed in [10] is used here. Note that at this time, each core stack is treated as a core in 2-D. Basically, the heuristic algorithm first sorts the core stacks in a descending order given the total communication quantities between this stack and other ones. Then the core stack with largest communication quantity is mapped onto an empty tile in NoC platform with the maximum neighbor tiles, and the neighbor tiles are flagged with the same V/F as the mapped cores. During the mapping process, if V/F of the core being mapped is equal to the V/F of the cores which have been mapped, the core stack will be mapped to any candidate tiles with same V/F. This can ensure that the cores with same V/F can be grouped into one VFI as much as possible. Above process repeats until all the core stacks have been mapped.

V. EXPERIMENT

A. Experimental Setup

Table 1. Statistic of task graphs

Task graph	Task#	Communication	in/out degree
TG1	88	low	1.2/1.2
TG2	86	low	1.2/1.2
TG3	92	medium	1.5/1.5
TG4	98	medium	1.5/1.5
TG5	96	medium	2/2
TG6	100	medium	2/2
TG7	101	high	2/2
TG8	102	high	2/2
TG9	100	high	2/2
TG10	102	high	2/2

Simulation platform: experiments are performed on a hypothetical 3-D multi-core SoC which is tile-based NoC-Bus structure with $4 \times 4 \times 2$ topology. The core is similar to the VLIW processor in TILE64 [21] with the nominal operating frequency of 500MHz under the supply voltage of 1.0V. The core is assumed to have five available voltage levels [0.7V,0.8V,0.9V,1.0V,1.1V]. Mixed-voltage/mixed-frequency FIFOs are used at border between VFIs for data synchronization.

Task graph: totally 10 task graphs are generated using TGFF [22], each of which includes a number of tasks ranging from 80 to 100. Moreover, in/out degrees of the task as well as the communication quantity are changed in the task graph generation to cover various task categories. Table 1 lists the related statistic for the task graphs.

Thermal evaluation: HotSpot 5.0 is exploited for temperature calculation [24]. The configuration parameters for thermal simulation are as in [18] by extrapolating from 90nm process. Steady temperatures of the cores and peak temperature of the chip are calculated by feeding the power traces into HotSpot.

B. Experimental Results

Now we present the experimental results to illustrate the effectiveness of our synthesis framework on energy and thermal optimizations. For comparison purpose, we modified a thermal balancing algorithm proposed in [15], enforcing it with VFI configurations. Moreover, we also implemented an energy aware task scheduling and VFI partition scheme proposed in [7]. We call them as *TB&VFI* and *EAS* in the rest of the paper. Both of *TB&VFI* and *EAS* schemes are originally applied to the hard NoC platform. The former schedules the tasks onto the cores by considering the inherent thermal variation within and across the tasks. While the latter combines task scheduling and VFI partition to minimize system energy. In the experiments, both of them are assumed to conduct on hard 3-D NoC platform with same hardware configurations to our scheme. As for VFI partition, in *EAS* and our framework, number of VFIs are determined by the V/F scaling results and the underlying VFI partitioning algorithms. *TB&VFI* originally does not implement VFI partition. Hence in the experiments, VFI partition in *TB&VFI* is as same as the one in *EAS*. Table 2 illustrates VFI partitioning results for three schemes.

Table 2. Statistic on VFI partition

	TB&VFI	*EAS*	Ours
nVFI	8	8	6
nCore	3-6	3-6	4-6

nVFI: number of VFIs; nCore: number of cores in VFIs

B.1 Energy Optimization Results

Figure 5 illustrates energy reduction and peak temperature after applying *TB&VFI*, *EAS* and our schemes on 10 task graphs (from TG1 to TG10). For clarity, energy optimization results are normalized to the ones obtained from *TB&VFI*. As shown in Figure 5(a), our synthesis framework outperforms much better than *TB&VFI* on energy reduction. On average energy reduction of 18.6% is achieved by our framework over the one in *TB&VFI*. Moreover, observing Figure 5(b), our framework also produces the lower peak temperature than *TB&VFI*. On the other hand, with comparable results on energy reduction, the resulted peak temperatures in our framework for executing all the task graphs are much lower than the one in *EAS*. On average reduction in peak temperature is reached about 5.7 ℃, compared with *EAS*.

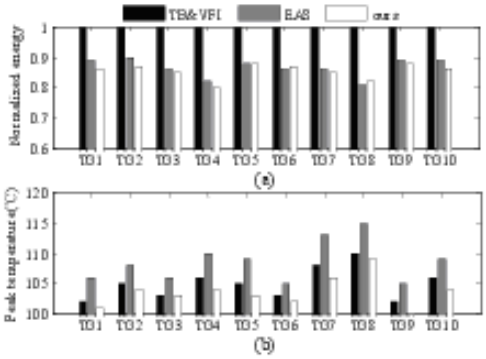

Figure 5. Resulted Energy reduction and peak temperature, (a) energy reduction and (b) peak temperature.

B.2 Thermal Balancing Results

Let's check the capability of three schemes on balancing the thermal across the chip. In this time all the ten task graphs are continuously executed one after the other. We uniformly divide the total execution time of each task graph into 10 time segments and collect the power traces in all of the 100 time segments. Then the obtained power trace file is fed into HotSpot for calculating the average temperatures for all the core stacks, which are as shown in Figure 6. Note that due to the control dependency between the tasks, only a few of the tasks can be executed at the beginning of executing the first task graph (TG1) and at end of executing the last task graph (TG10). As a result, large temperature variations exist at such time since most of the core stacks have not been assigned with the tasks. Given that, the output temperatures from HotSpot in a little of time durations at the beginning of executing TG1 (about 15ms) and at the end of executing TG10 (about 10ms) are removed from the total experimental

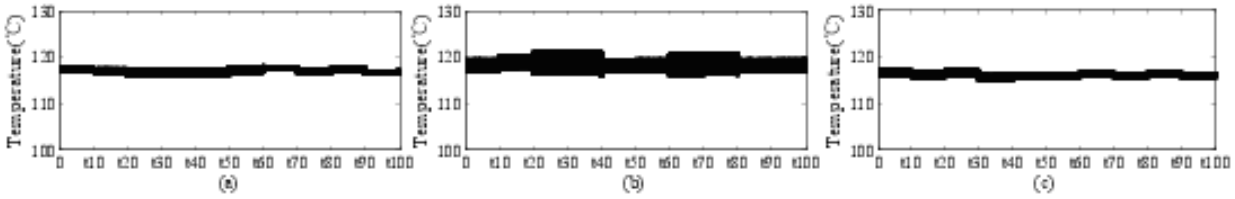

Figure 6. Average temperatures across 16 core stacks under executing 10 task graphs, (a) *TB&VFI*, (b) *EAS,* and (c) ours.

results. Another thing should be noticed is that the temperature resulted from continuously executing 10 task graphs is a little higher than separately executing individual task graphs. Therefore, temperature presented in Figure 6 is higher than the one in Figure 5(b).

VI. CONCLUSIONS

In this paper, we proposed to synthesis 3-D NoC with VFI design for minimizing system energy meanwhile keeping thermal balancing across the chip. Unified considering task scheduling and V/F scaling help to minimize the computation energy. Core stacking following with task migration can reduce communication energy while balancing the powers across the stacks. Moreover, treating each core stack as a unity facilitates to transform the 3-D NoC mapping problem into a 2-D issue. Experimental results demonstrate the effectiveness of the proposed synthesis framework.

VII. ACKNOWLEDGE

The work was supported in part by National Natural Science Foundation of China (NSFC) under grant No.(61204027, 61306049), in part by Natural Science Foundation of Hebei province of China under grant No. F2013502274, in part by Natural Science Foundation of Anhui Province under No.1208085QF127, in part by the Fundamental Research Funds for the Central Universities under Grant No.(12MS123,13MS69).

REFERENCES

[1] S. Borkar. 3D Integration for Energy Efficient System Design. ACM/IEEE Design Automation Conference (DAC), pp.214-219, 2011.

[2] F. Clermidy, F. Darve, D. Dutoit et al. 3D Embedded Multi-core: Some Perspectives. ACM/IEEE Design, Automation and Test in Europe (DATE), pp.1327-1332, 2011.

[3] S. Herbert, D. Marculescu. Characterizing Chip-Multiprocessor Variability-Tolerance. ACM/IEEE Design Automation Conference (DAC), pp.313-318, 2008.

[4] J. Dorsey, S. Searles, M. Ciraula et al. An Integrated Quad-core Opteron Processor. IEEE International Solid-State Circuits Conference (ISSCC), pp.102-103, 2007.

[5] J. Howard, S. Dighe, S. R. Vangal et al. A 48-Core IA-32 Processor in 45 nm CMOS Using On-Die Message-Passing and DVFS for Performance and Power Scaling. IEEE Journal of Solid-State Circuits (JSSC), Vol.46, No.1, pp.173-183, 2011.

[6] W. Hung, G. M. Link, Y. Xie et al.Interconnect and Thermal-Aware Floorplanning for 3D Microprocessors. IEEE International Symposium on Quality Electronic Design (ISQED), pp.98-104, 2006.

[7] U. Y. Ogras, R. Marculescu, P. Choudhary et al. Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip. ACM/IEEE Design Automation Conference (DAC), pp.110-115, 2007.

[8] J. Hu, R. Marculescu. Communication and Task Scheduling of Application-specific Net120works-on-chip. Computers and Digital Techniques, Vol.152, No.5, pp.643-651, 2005.

[9] Y. Zhang, S. Hu, Z. Danny. Task Scheduling and Voltage Selection for Energy Minimization. ACM/IEEE Design Automation Conference (DAC), pp.183-188, 2002.

[10] W. Jang, D. Duo, D. Z. Pan. A Voltage-Frequency Island Aware Energy Optimization Framework for Networks-on-chip. ACM/IEEE International Conference on Computer-Aided Design (ICCAD), pp.264-269, 2008.

[11] B. Goplen, S. Sapatnekar. Thermal Via Placement in 3D ICs. IEEE International Symposium on Physical Design (ISPD), pp.167-174, 2005.

[12] E. Wong, S. K. Lim. 3D Floorplanning with Thermal Vias. ACM/IEEE Design, Automation and Test in Europe (DATE), pp.1-6, 2006.

[13] M. S. Bakir, C. King, D. Sekar et al. 3D Heterogeneous Integrated Systems: Liquid Cooling, Power Delivery, and Implementation. IEEE Custom Integrated Circuits Conference (CICC), pp.663-670, 2008.

[14] A. K. Coskun, J. L. Ayala, D. Atienza et al. Dynamic Thermal Management in 3D Multicore Architectures. ACM/IEEE Design, Automation and Test in Europe (DATE), pp.1410-1415, 2009.

[15] X. Zhou, J. Yang, Y. Xu et al. Thermal-aware Task Scheduling for 3D Multicore Processors. IEEE Transactions on Parallel Distribution System (TPDS), Vol.21, No.1, pp.60-71, 2010.

[16] C. Zhu, Z. Gu, L. Shang et al. Three-dimensional Chip-multiprocessor Run-time Thermal Management. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol.27, No.8, pp.1479-1492, 2008.

[17] U. Y. Ogras, J. Hu, R. Marculescu. Key Research Problems in NoC Design: A Holistic Perspective. ACM/IEEE International conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp.69-74, 2005.

[18] Y. Cheng, L. Zhang, Y. Han et al. Thermal-constrained Task Allocation for Interconnect Energy Reduction in 3-D Homogeneous MPSoCs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.21, No.2, pp.239-249, 2013.

[19] K. Kang, J. Kim, S. Yoo et al. Runtime Power Management of 3-D Multi-Core Architectures Under Peak Power and Temperature Constraints. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.30, No.6, pp.905-918, 2010.

[20] F. Li, C. Nicopoulos, T. Richardson et al. Design and Management of 3D Chip Multiprocessors UsingNetwork-in-memory. ACM/IEEE International Symposium on Computer Archetecture (ISCA), pp.130-141, 2006.

[21] B. Shane, E. Bruce, A. John et al. TILE64 Processor: A 64-Core SoC with Mesh Interconnect. IEEE International Solid-State Circuits Conference (ISSCC), pp.88-598, 2008.

[22] Available: http://ziyang.eecs.umich.edu/~dickrp/tgff/

[23] G. M. Link, N. Vijaykrishnan. Thermal Trends in Emerging Technologies. International Symposium on Quality Electronic Design (ISQED), pp.8-632, 2006.