TAPP: Temperature-Aware Application Mapping for NoC-Based Many-Core Processors

Di Zhu†, Lizhong Chen‡, Timothy M. Pinkston†, and Massoud Pedram†
†Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, United States
‡School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, United States
Email: †{dizhu, tpink, pedram}@usc.edu, ‡chenliz@eecs.oregonstate.edu

***Abstract*—Application mapping with its ability to spread out high-power components can potentially be a good approach to mitigate the looming issue of hotspots in many-core processors. However, very few works have explored effective ways of making tradeoff between temperature and network latency. Moreover, on-chip routers, which are of high power density and may lead to hotspots, are not considered in these works. In this paper, we propose *TAPP (Temperature-Aware Partitioning and Placement)*, an efficient application mapping algorithm to reduce on-chip hotspots while sacrificing little network performance. This algorithm "spreads" high-power cores and routers across the chip by performing hierarchical bi-partitioning of the cores and concurrently conducting placement of the cores onto tiles, and achieves high efficiency and superior scalability. Simulation results show that the proposed algorithm reduces the temperature by up to 6.80°C with minimal latency increase compared to the latency-oriented mapping solution.**

I. Introduction

With continued technology scaling, the number of processing cores on-chip has been growing to tens or even a hundred [5][8][19]. Consequently, networks-on-chip (NoCs) have been adopted as the *de facto* architecture in large application-specific multiprocessor systems-on-chips (MPSoCs) and general-purpose chip-multiprocessors (CMPs) to support communication of many concurrently running threads. In these NoC-based many-core systems, an important design and run-time step is application mapping, which maps the threads of application(s) to distributed tiles. Since application mapping affects nearly all aspects of on-chip communication, it is critical to the efficacy of NoC as well as the overall many-core system.

Application mapping has drawn great attention over the past decade and has been optimized for various objectives such as network latency, power, energy, cost, fairness (to list a few, [7][9][13][20]). However, very few works have considered application mapping with the awareness of chip temperature (to our knowledge, only [1][10][15] do this), despite the large impact of mapping on the chip temperature and the impact of temperature on performance, reliability, lifetime, and leakage power dissipation of the chip [16]. Moreover, none of these works take into account the thermal effects of on-chip routers. That is, only the core power consumption is accounted for as a heat source in those mapping schemes, leading to inaccurate and possibly erroneous hotspot estimation. In fact, as shown in Section III, NoC routers, which have relatively small chip area but large power consumption, can potentially become hotspots themselves. In addition, existing temperature-aware mapping schemes all use search-based algorithms, which are very slow to generate satisfying mapping results under execution time constraints, especially if runtime mapping adjustments are needed.

In this paper, we address the imperative issue of temperature-aware application mapping with the consideration of NoC router power. A fast yet effective mapping algorithm, called *TAPP (Temperature-Aware Partitioning and Placement)*, is proposed. The key challenge is to balance the potential conflict between the chip temperature and the network latency. The basic idea of TAPP is to "spread" core and router power by hierarchical bi-partitioning while concurrently performing partial block placement in each partitioning iteration. Both core and NoC router power consumption, as well as network latency factor, are reflected in the cost function of min-cut in partitioning and the cost function in placement.

The main contributions of this paper are the following. We (i) analyze the importance of taking into account the NoC router power in application mapping, (ii) formulate the problem of temperature-aware application mapping that considers the thermal effects of both cores and the NoC routers, and (iii) propose an efficient heuristic-based algorithm with $O(N^3)$ time complexity (N is the network size), suitable for both design-time static mapping and run-time dynamic mapping.

II. Background and Motivation

A. Temperature Model

A common approach to accurately capture the steady-state thermal distribution while requiring limited input parameters is the thermal resistance model [12]. To facilitate the modeling of tile-based many-core chips, we consider each tile as one power source, whose power is the summation of the power consumption of the core and the co-located router on that tile.

The temperature increase of a node with coordinate (x, y) (i.e., a tile in the context of this work) is the summation of the temperature increase introduced by the power source on this node and on every other power sources:

$$T(x, y) = T_0 + \sum_{i=1}^{N} \Delta T_i. \qquad (1)$$

The temperature increase ΔT_i by the power source at tile i is a function of the power consumption P_i of the source and the distance r_i between the node at (x, y) and tile i, i.e.,

$$\Delta T_i = P_i \cdot \Delta T^U(r_i) \qquad (2)$$

where $\Delta T^U(r_i)$ is the temperature increase caused by unit power of a heat source, and it only depends on the Euclidean distance r_i between the node in question and the source. According to curve-fitting HotSpot [17] simulation results, it is subject to exponential decay with r_i.

B. Packet Latency Model

The packet latency model is derived based on [4][20] to calculate the on-chip network latency $L_{i,j}$ of a packet generated at the i-th tile and heading for the j-th tile, i.e.,

$$L_{i,j} = \left(H_{i,j} + 1\right) \times \left(td_r + td_q\right) + H_{i,j} td_l + td_s, \qquad (3)$$

where $H_{i,j}$ is the number of hops between tile i and tile j, which is the Manhattan distance for mesh network with XY routing. td_r, td_l, td_q, and td_s are the per-hop latency for router and link, per-router queuing latency (0~1 cycles as observed in the simulation), and serialization latency (pre-determined for a given packet format and NoC architecture), respectively.

C. Thermal Impact of NoC Routers

NoC routers may have large impact on thermal issue due to their higher power-to-area ratio compared to other on-chip components. To illustrate, Figure 1 plots this ratio for the main components in Scorpio, a newly fabricated 36-core CMP [5]. With 10% of chip's area and 19%

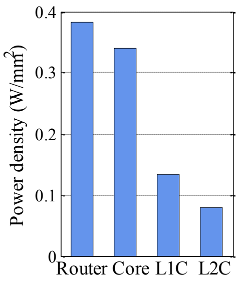

Figure 1. Power densities of different components on a tile in Scorpio.

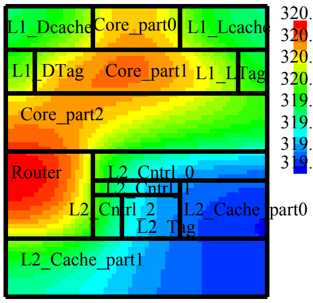

Figure 2. Thermal map of one tile in Scorpio (the core has three parts due to the rectangular restriction in HotSpot).

of chip's power, NoC has the highest power-to-area ratio. Figure 2 is the thermal map of Scorpio by inputting the key chip parameters to Hotspot [17]. The figure shows that NoC component can potentially be the hotspot in each tile. However, reducing NoC temperature in application mapping requires placing high-power cores far from each other, whereas the goal of reducing on-chip latency may require these active cores as close as possible. To address this issue, new temperature-aware mapping scheme is much needed, as proposed in this work.

III. Problem Formulation

Suppose a NoC-based system has N tiles, and a set of N threads (cores) to be mapped onto the chip. We define the thread communication graph (TG) as follows: within the TG, a vertex i where $i \in \{1,2,\dots,N\}$ indicates one thread, its weight P_i is the power consumption of this thread, and the directed edge from thread i to j has the weight $c_{i,j}$ representing communication rate, i.e., the average number of flits sent from thread i to thread j per unit time length.

Another directed graph, the tile latency graph (LG), has each of its vertex indicating one tile on the chip, and the edge from tile i to tile j has the weight $l_{i,j}$ indicating the average packet latency of sending one flit from tile i to j. A router is affixed to each tile, and its power consumption is calculated by

$$P_R = P_{R,sta} + P_{R,dyn}(C_R) \tag{4}$$

where $P_{R,sta}$ is the static power consumption of the router, and $P_{R,dyn}$ is the dynamic power consumption which is a function of the traffic going through this router C_R (in flits per cycle), calculated by the sum of $c_{i,j}$ where the communication between core i and j goes through this router. This relies on the locations of core i and j. Therefore, C_R is dependent on the core mapping results.

The goal of application mapping is to find a permutation

$$j \to \pi(j) \in \{1,2,\dots,N\}, j \in \{1,2,\dots,N\} \tag{5}$$

where $j \to \pi(j)$ denotes that the j-th core is mapped onto the $\pi(j)$-th tile.

As network latency is among the most important criteria of on-chip networks, the temperature-aware mapping methods should ensure as little latency increase as possible while reducing on-chip temperature. The overall average packet latency L and the maximum on-chip temperature T_M are calculated by

$$L = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} c_{i,j} l_{\pi(i),\pi(j)}}{\sum_{i=1}^{N}\sum_{j=1}^{N} c_{i,j}}, \tag{6}$$

$$T_M = \max_i T\left(x_{\pi(i)}, y_{\pi(i)}\right) \tag{7}$$

where $x_{\pi(i)}, y_{\pi(i)}$ is the coordinates of tile $\pi(i)$.

Therefore, the optimization goal of the proposed temperature-aware mapping algorithm is the balance between the maximum on-chip temperature T_M and the overall average packet latency L,

$$\varphi \cdot L + \psi \cdot T_M \tag{8}$$

The units of coefficients φ and ψ are cycle^{-1} and K^{-1}, respectively. The objective function can be adjusted by varying these two coefficients.

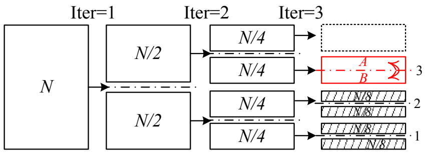

Figure 3. Step 1.

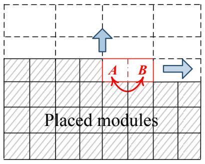

Figure 4. Step 2.

Unfortunately, this optimization problem is hard. Even with $\psi = 0$ in the formulated problem (latency-minimizing mapping), this simplified version has the form of a *Quadratic Assignment Problem* (*QAP*), which is *NP-hard*. Thus, the scalability of exact algorithms is greatly restricted due to their high complexity, and an efficient heuristic algorithm is needed.

IV. Proposed Solution

A. Motivation

In traditional circuit partitioning algorithms, the primary goal is to find the min-cut of a graph of gates, i.e., to minimize the total weight of interconnections that cross the cutline. We apply similar approach to application mapping based on Kernighan–Lin (KL) algorithm [11], considering two factors in the weight of interconnection. First, the weight needs to reflect the overall traffic (communication rate) between two partitions A and B, i.e., $C_{AB}^{par} = \sum_{i\in A, j\in B} c_{ij}$. Since higher power density regions usually lead to hotspots, the on-chip power consumption should be evenly spread out in order to reduce the maximum chip temperature. Therefore, we add the difference in power consumption between two partitions to the total cost C_{AB}^{par} of a cutline, the second factor in addition to the traffic cost. In this way, minimizing C_{AB}^{par} means concurrently balancing power and minimizing latency during partitioning.

Specifically, the partition cost of a cutline C_{AB}^{par} which divides the current graph into block A and B is calculated by

$$C_{AB}^{par} = \hat{\varphi} \cdot \sum_{i\in A, j\in B} c_{ij} + \hat{\psi} \cdot \left| \sum_{i\in A} \hat{P}_i - \sum_{j\in B} \hat{P}_j \right| \tag{9}$$

where the power $\hat{P}_i$ of each tile i on a partition is calculated as the sum of the core power P_i, the NoC router static power and the part of router dynamic power caused by all the communication generated at or destined for this tile. Note that the part of router dynamic power caused by forwarding bypassing traffic is not included as it depends on the mapping results and is not reflected at this step. It will be accounted for in the final temperature and latency calculation.

The above equation (9) and the min-cut approach provide a quasi means of performing tradeoff in equation (8): increase $\hat{\varphi}/\hat{\psi}$ value towards more latency-oriented mapping, and decrease $\hat{\varphi}/\hat{\psi}$ towards more temperature-oriented mapping.

B. Temperature-Aware Partition-and-Placement (TAPP)

The key idea behind the proposed TAPP is that the block placement is carried out *concurrently* within each iteration of the hierarchical bi-partitioning. TAPP consists of three steps.

1) Step 1: Horizontal Partition-and-Placement.

The first step is to conduct horizontal partitioning with placement until the size of each partition equals one row in the mesh network, based on the partitioning cost calculation C_{AB}^{par} in (9). Each iteration includes the partitioning and placement of all the current blocks, so the number of blocks doubles after each iteration, as shown in Figure 3. As soon as the min-cut partitioning is finished in each partition, the placement order of the two blocks is determined right away.

Take the third iteration as an example, where the size of each block after partitioning is $N/8$. Suppose the partition and placement is performed from the bottom to the top, and the first and second partition-and-placement iterations have been finished (the shadowed four blocks in Figure 3). The third block is partitioned into two blocks, A and B. Whether A or B is placed on top is determined by the costs of these two placement options. For example, the cost of A placed on top $C^{pl}_{A>B}$ is the sum of the communication cost and the temperature increase caused by A and B, comprised of A on top $C^{pl}_{A>B,A}$ and B at bottom $C^{pl}_{A>B,B}$. The former can be calculated by the following equation (note that this cost in placement is different from the partitioning cost used in min-cut):

$$C^{pl}_{A>B,A} = \sum_{F} \left(\hat{\varphi} \cdot c_{A,F} M_{A,F} + \hat{\psi} \cdot \hat{P}_A \cdot \Delta T^U\left(r_{A,F}\right)\right) \qquad (10)$$

where $c_{A,F} = \sum_{i\in A, j\in F} c_{ij}$ is the total communication rate between block A and a placed block F(e.g., a shaded block), $M_{A,F}$ and $r_{A,F}$ are the Manhattan distance and Euclidean distance between the *centers* of A and F. This is because the tile assignment of each individual core might not be determined at this stage, so we assume all the cores within each block are placed at the center. Similarly we calculate the B part $C^{pl}_{A>B,B}$, sum it up with $C^{pl}_{A>B,A}$ to get $C^{pl}_{A>B}$, and then compare it with the cost of B placed on top $C^{pl}_{B>A}$.

The horizontal partition-and-placement in Step 1 has $\log_2 N$ iterations, each iteration containing 2^{k-1} times of KL partition algorithm. The evaluation for placement after each partitioning takes $O(\sqrt{N})$ because the number of rows is $O(\sqrt{N})$. Since the time complexity of KL is $O(N^3)$ for a graph of size N [11], the time complexity of Step 1 is calculated by

$$\sum_{k=1}^{\log_2 N} 2^{k-1} \cdot \left(O\left(\left(\frac{N}{2^{k-1}}\right)^3\right) + O(\sqrt{N})\right) = O(N^3) \qquad (11)$$

2) Step 2: Vertical Partition-and-Placement

With the cores assigned to each row determined after the first step, we perform the vertical bi-partitioning with placement hierarchically until each block contains one core.

In each iteration, the bi-partitioning is performed bottom to top, and left to right, as shown in Figure 4. Each of the current blocks is partitioned with minimum cost as in (9), and the order of the two blocks after partitioning is calculated similarly as in the horizontal placement. For example in Figure 4, after the partitioning of the A and B block, we calculate the temperature and latency costs of all the placed blocks (including the placed blocks in the same row) caused by A and B in the two placing ways, i.e., either A on the left or B on the left, and choose the one with smaller cost.

Step 2 performs KL hierarchically for each row with size $\sqrt{N}$ similar to in Step 1, but the placement evaluation takes $O(N)$ instead of $O(\sqrt{N})$, therefore the time complexity is $\sqrt{N} \cdot \left(O\left(\sqrt{N}^3\right) + O(N) \cdot O(N)\right) = O(N^{2.5})$.

3) Step 3: Local Adjustment

Before reaching the final placement solution, we conduct local adjustment to the assignment generated in Step 2 as fine tuning. A 2x2 window is slid from bottom left to top right, and the 24 permutations of the four cores inside the window are evaluated, and we greedily pick the permutation with minimum cost to assign them to the four tiles, with $O(N^2)$ complexity.

To sum up, the timing complexity of TAPP is $O(N^3)$.

C. Applicability in Run-Time Application Mapping

Run-time mapping can be much needed in many-core processors with dynamic workloads, for example, where new threads are allocated to replace the ones that have been finished while other unfinished threads continue running on certain tiles. Unlike the exiting time-consuming temperature-aware mapping algorithms, the proposed TAPP is applicable for dynamic mapping. Specifically, to solve the run-time mapping problem in TAPP, the running threads are considered as fixed nodes in the KL partition algorithm, and the order of the two partitioned blocks is also fixed if any one of them contains a fixed block.

V. Simulation Results

A. Simulation Setup

The proposed problem and schemes are evaluated with both CMP traces and MPSoC traces. The CMP traces are generated by the cycle-accurate gem5 simulator [3] and the McPAT power modeling framework [14], with the multi-threaded PARSEC benchmarks [2]. The MPSoC task graphs and power traces are generated by TGFF [6]. The interconnection power consumption is calculated by DSENT [18].

We have the following five schemes.

1. Random, the average latency and temperature of randomly generated mapping results;
2. MinLat_SA, a simulated annealing algorithm aiming at minimizing the overall average packet latency, as the baseline algorithm;
3. CoreOnly_SA, a temperature-aware simulated annealing algorithm which considers only core power for generating mapping (the final temperature calculation includes routers);
4. CoreNoC_SA, a temperature-aware simulated annealing algorithm solving the proposed temperature-aware mapping problem, i.e., considering both core power and NoC power;
5. CoreNoC_TAPP, the proposed heuristic algorithm to solve the temperature-aware mapping problem.

B. Temperature and Latency Results

We test the five schemes on an 8x8 mesh network with grid size = 1 mm. For the CMP traces, we use four groups P1, P2, P3, and P4 as listed in Table I, each of them containing four 16-thread PARSEC benchmarks. For the MPSoC traces, we use TGFF to generate four application graphs for the 64 cores to be placed, as listed in Table II. The wide range of configurations in these two tables show a representative set of workloads.

The CMP results and MPSoC results are plotted in Figure 5 as the tradeoff curves of temperature and latency. The latency achieved by Random is around 28.7 cycles for CMP and 30.7 cycles for MPSoC, and the Random temperature result is shown as text. Note that the CMP results show less than 20 cycles in delay because the actual communication graph of each of the four groups is comprised of four disconnected subgraphs due to the lack of inter-application communication.

We have the following three observations.

First, it is noted that the MinLat_SA which solely minimizes the average network latency reduces the on-chip temperature compared to Random mapping result because the MinLat_SA greatly reduces the NoC communication power.

Second, the CoreOnly_SA, which only considers the core power consumption during mapping, decreases the hotspot temperature, but still averagely 1-2°C higher than the two solutions with NoC power-awareness. This demonstrates that the proposed problem which takes into account the NoC power is a better description of real thermal distributions on chip.

Table I. PARSEC benchmark configurations.

No.	**PARSEC Benchmarks**	**P_i Avg**	**P_i Std dev**
P1	blackscholes, bodytrack, canneal, ferret	0.4564	0.1678
P2	bodytrack, canneal, ferret, vips	0.5036	0.1028
P3	blackscholes, dedup, ferret, fluidanimate	0.3938	0.1779
P4	canneal, fluidanimate, swaptions, x264	0.3717	0.0757

Table II. TGFF benchmark configurations.

Testbench	**P_i Avg**	**P_i Std dev**
tgff1	0.8346	0.4036
tgff2	0.4259	0.3027
tgff3	0.8130	0.1514
tgff4	0.2502	0.1010

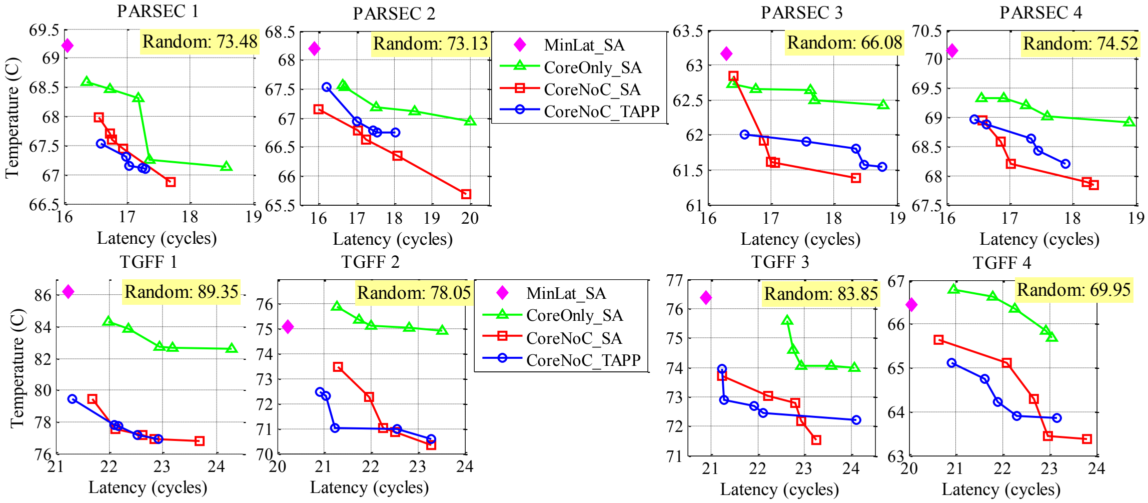

Figure 5. PARSEC benchmark results (average latency of Random: 28.7 cycles) and TGFF benchmark results (average latency of Random: 30.7 cycles).

Third, the CoreNoC_SA and TAPP achieve averagely equal results. For the four CMP benchmarks, CoreNoC_SA and TAPP reduce the maximum temperature by 1.36°C and 1.25°C with averagely 2.97% and 2.17% latency penalty compared to MinLat_SA algorithm, respectively. The maximum penalty of TAPP is 3.30% in PARSEC 1. For the four MPSoC benchmarks, they reduce temperature by 3.66°C and 4.40°C with averagely 1.87% and 2.32% latency penalty. The maximum penalty of TAPP is 3.40% in TGFF 2. However, the results achieved by TAPP are obtained within a much smaller execution time. The SA results shown in Figure 5 are given 1e5 times of iterations each, taking an average of 437.4 seconds, leading to 112X execution time compared to TAPP which takes 3.9 seconds to finish.

C. *Power Consumption*

The power overhead associated with the temperature-aware mapping in previous and this work comes from the extra dynamic power of NoC due to higher network activity. However, since the TAPP algorithm achieves minimal latency penalty, the activity factor of NoC is increased only by a small amount, thus introducing very small power overhead. According to McPAT and DSENT models, TAPP has an average chip power overhead of 0.21% for PARSEC benchmarks and 0.96% for TGFF benchmarks.

VI. Conclusions

This paper addresses the important issue of temperature-aware application mapping in many-core processors. We analyze the thermal impact of NoC routers and formulate a mapping problem that takes into consideration the power of cores and routers as well as the tradeoff between latency and temperature. An efficient temperature-aware partitioning and placement (TAPP) algorithm is proposed to mitigate on-chip hotspots while sacrificing little network performance. Simulation results on 8x8 mesh networks using PARSEC and TGFF benchmarks demonstrate the effectiveness of TAPP in mapping results and algorithm execution time.

VII. Acknowledgement

We sincerely thank Dr. Zhiliang Qian and the anonymous reviewers for their helpful comments and suggestions. This research is supported by the National Science Foundation (NSF) grant CCF-1321131, Software and Hardware Foundations of the NSF and the Semiconductor Research Corporation.

References

[1] C. Addo-Quaye, "Thermal-aware mapping and placement for 3-D NoC designs." SOC Conference, 2005. IEEE, 2005.

[2] C. Bienia, and L. Kai. "Parsec 2.0: A new benchmark suite for chip-multiprocessors." Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation. 2009.

[3] N. Binkert, et al. "The gem5 simulator." ACM SIGARCH Computer Architecture News 39.2, 2011.

[4] W. J. Dally and B. Towles, "Principles and practices of inter-connection networks," Morgan Kaufmann, 2003.

[5] B.K. Daya, et al. "SCORPIO: a 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering." International Symposium on Computer Architecture. IEEE, 2014.

[6] R. P. Dick, D. L. Rhodes, & W. Wolf, "TGFF: task graphs for free." Proceedings of the 6th international workshop on Hardware/software codesign, pp. 97-101. IEEE, 1998.

[7] A. Faruque, et al. "ADAM: run-time agent-based distributed application mapping for on-chip communication." Proceedings of the 45th annual Design Automation Conference. ACM, 2008.

[8] J. Howard, et al., "A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," Proceedings of the Solid-State Circuits Conference Digest of Technical Papers, 2010.

[9] J. Hu, and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," Proceedings of the Asia and South Pacific Design Automation Conference, 2003

[10] W. Hung, et al. "Thermal-aware IP virtualization and placement for networks-on-chip architecture." Proceedings of International Conference on Computer Design. IEEE, 2004.

[11] B. W. Kernighan, & S. Lin. An efficient heuristic procedure for partitioning graphs. Bell Systems Technical Journal, 1970.

[12] F. Kreith. The CRC handbook of thermal engineering. Springer, 2000.

[13] S. Murali, and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," Design, automation and test in Europe, 2004.

[14] S. Li, et al. "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures." Microarchitecture, IEEE/ACM International Symposium on. IEEE, 2009.

[15] Y. Liu, Y. Ruan, Z. Lai, & W. Jing. "Energy and thermal aware mapping for mesh-based NoC architectures using multi-objective ant colony algorithm." International Conference on Computer Research and Development. IEEE, 2011.

[16] L. Shang, L. S. Peh, A. Kumar, & N. K. Jha, "Thermal modeling, characterization and management of on-chip networks." Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 2004.

[17] K. Skadron, et al. "Temperature-aware microarchitecture." ACM SIGARCH Computer Architecture News. Vol. 31. No. 2. ACM, 2003.

[18] Sun, C., et al. (2012). DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling. In International Symposium on Networks-on-Chip.

[19] S. Vangal, et al. "An 80-tile 1.28 TFLOPS network-on-chip in 65nm CMOS." International Solid-State Circuits Conference. IEEE, 2007.

[20] D. Zhu, et al. "Balancing On-Chip Network Latency in Multi-Application Mapping for Chip-Multiprocessors." IPDPS 2014.