Thermal-aware Task Scheduling for 3D-Network-on-Chip: A Bottom-to-Top Scheme

Yingnan Cui[1], Wei Zhang[2], Vivek Chaturvedi[1], Weichen Liu[3], and Bingsheng He[1]

[1]School of Computer Engineering, Nanyang Technological Univ., Singapore, {ycui1,vchaturvedi,bshe}@ntu.edu.sg
[2]Department of Electronic & Computer Engineering, Hong Kong Univ. of Sci. and Tech., Hong Kong, wei.zhang@ust.hk
[3]College of Computer Science, Chongqing University, Chongqing, China, wliu@cqu.edu.cn

***Abstract*—3D-NoC emerges as a potential multi-core architecture delivering high performance, high energy efficiency and great scalability. However, severe thermal challenges due to high power density continue to be a major hurdle in the development of 3D-NoC. In this paper, we propose a novel thermal-aware task scheduling scheme named Bottom-to-Top (B2T) approach to address this challenge. Incorporating communication overhead into analysis based on task graph, this heuristic-based method judiciously performs task allocation on processing units to efficiently minimize the peak temperature and improve the execution time of the applications. Our simulation results demonstrate that our scheme can achieve significant peak temperature reduction (up to $7.95°C$) and performance improvement (up to 4%) when compared to other methods.**

I. Introduction

In recent years, 3D-Network-on-Chip (3D-NoC) has become a promising architecture for large-scale multiprocessors due to its high scalability, high inter-processor communication bandwidth and low area cost [1]. However, when compared to its 2D counterpart, thermal related side effects in 3D-NoC have drawn increasing concerns due to the significant increase in power density [2]. Therefore, efficient thermal management is crucial for 3D-NoC [3].

Among various thermal management methods, thermal-aware task scheduling is a light-weight, highly efficient technique [4]. Previous works like [5], [6], [7] discussed OS level thermal-aware scheduling for multiprocessors. For higher throughput, embedded system applications such as multi-media applications are usually modeled as task graphs, and the inter-dependence among tasks can be further exploited to improve the thermal efficiency.

Works in [8], [9] formulated the thermal-aware task graph scheduling problem as a formal optimization problem and used methods like exhaustive search and integer linear programming algorithms to find the optimal solution. The high computation complexity of these works prohibits their use in run-time cases. Authors in [10] and [11] presented heuristic-based solutions to the thermal-aware task graph scheduling problems. However, the heuristic used in these algorithms are for 2D-NoC and does not differentiate the processors on different layers. Directly applying such algorithms for 3D-NoC can not efficiently reduce the temperature of the system.

[12] points out that the temperature of 3D-NoC is closely related to the power distribution in vertically stacked processors. By placing high power tasks on layers which are closer to the heatsink, the temperature of the 3D-NoC system could be effectively reduced. Based on this observation, we propose an efficient thermal-aware task scheduling algorithm for 3D-NoC which is called Bottom-to-Top (B2T) scheme. The B2T scheme first schedules all the tasks to the bottom layer of a 3D-NoC to uniformly distribute power consumption among each stack of processors, and then moves low power tasks to the top layer to reduce the execution time by exploiting parallelisms of the tasks while maintaining low operating temperature. Our B2T scheme also takes inter-processor communication

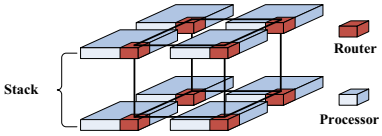

Fig. 1. Two-layer 3D-NoC architecture with 3D-mesh topology.

of 3D-NoC into consideration to further reduce execution time and power consumption of the system. Compared to previous thermal-aware scheduling algorithms designed for 2D-NoC, our B2T scheme significantly reduces the peak temperature with better or comparable performance. The experiments show up to $7.95°C$ peak temperature reduction together with up to 4% reduction in execution time when compared thermal-aware scheduling algorithms designed for 2D-NoC.

The rest of this paper is organized as follows: Section II analyzes the thermal model of 3D-NoC and explains the rationale of the Bottom-to-Top scheme; Section III presents formal definition of the problem; Section IV describes the B2T thermal-aware scheduling scheme in detail; Section V shows the experimental results; and finally, Section VI concludes the paper.

II. Thermal Features of 3D-NoC

Fig. 1 shows the architecture of a two-layer 3D-NoC with 3D-mesh topology. In the 3D-NoC, two layers of processors with their interconnection networks are stacked together. Communication channels for processors in adjacent layers are formed by Through-silicon-vias (TSV). To ease following discussions, we call the processors placed vertically on top of each other across all the layers as a *stack* of processors.

Fig. 2 is the cross-sectional view of a 3D-NoC system. Conventionally, the layer which lies next to the heatsink is called as the bottom layer and the layer which lies most distant from the heatsink is called the top layer. Fig. 2 shows the heat dissipating paths of the 3D-NoC through the heatsink. The heat produced in top layer is more difficult to dissipate, because the heat dissipation path of the top layer is longer than the bottom layer [13]. Therefore, in order to maintain low operation temperature of 3D-NoC, the power consumption of the top layer should be as low as possible.

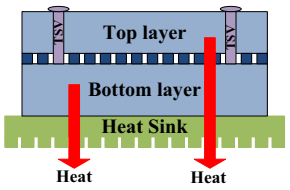

Fig. 2. Cross-sectional view of a stack of processors and heat dissipating path.

978-1-4799-4833-8/14/$31.00 ©2014 IEEE

TABLE I
SYMBOL DEFINITIONS

Symbols	Definition
$l(v)$	The length of the task v, measured in clock cycles.
$\epsilon(v)$	The energy consumed by task v.
$\rho(v)$	The power consumption of task v.
$w(e)$	The size of the message on edge e.
$\Phi(v)$	The priority of task v
P_b	Set of processors on the bottom layer of 3D-NoC
P_t	Set of processors on the top layer of 3D-NoC

III. PROBLEM FORMULATION

To ease the discussion of thermal-aware task scheduling algorithm in the next section, we introduce the definition of symbols and the problem formulation in this section.

In this paper, each application is denoted by a task graph; $G = (V, E)$, which is a directed acyclic graph, where V is the set of vertices representing the tasks and E is the set of edges representing the precedence relationship and communication between tasks. $l(v)$ is the execution time of task v, and $\epsilon(v)$ is the total energy consumed by task v. We assume that $l(v)$ and $\epsilon(v)$ are previously acquired knowledge. $\rho(v)$ is the power consumption of v, which is computed by $\epsilon(v)/l(v)$.

A 3D-NoC contains a set of processors $P = \{p\}$. Here we assume that the processors are homogeneous. A scheduling of the application is composed of a mapping function $M : V \mapsto P$ maps each task v to a processor p, which means task v will be executed on p; and a scheduling function $S : V \mapsto N$, which defines the execution order of the tasks on each processor. The objective of thermal-aware scheduling problem for 3D-NoC is to find a schedule for a given $G(V, E)$ and P, such that the peak operating temperature is reduced while the performance of the application is improved. Table I summarizes the definition of symbols used in this paper.

IV. THERMAL-AWARE TASK SCHEDULING

Our Bottom-to-Top (B2T) thermal-aware task scheduling scheme for 3D-NoC has two steps. The first step is a thermal-aware task mapping algorithm on bottom layer which maps the tasks of an application to the processors on the bottom layer with two objectives: firstly, the algorithm balances the power consumption of the tasks applied on each processor to avoid thermal hotspots; and secondly, the algorithm uses a critical path based scheduling method to reduce the execution time. The second step is an in-stack task adjustment algorithm. In each stack of processors, the algorithm exploits the parallelism of the tasks and selectively moves some of the tasks to be executed in parallel on the top layer processors to further reduce the execution time of the application while maintaining low operating temperature of the system unaffected. To help explaining the procedures of the B2T scheduling scheme, Fig. 3 shows an example of our schedule scheme which is used through this section. Fig. 3(a) shows a task graph waiting for scheduling; Fig. 3(b) shows a target 2-layer 3D-NoC system with four processor; Fig. 3(c) shows the mapping and scheduling results on the bottom layer; and Fig. 3(d) gives the final scheduling results after in-stack adjustment.

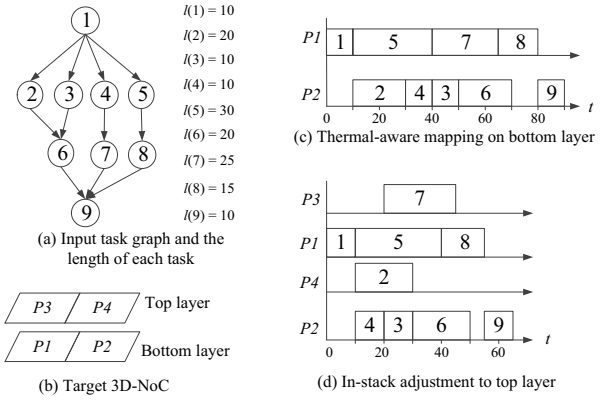

Fig. 3. A example of the two-step thermal-aware task graph scheduling algorithm.

A. Thermal-Aware Task Scheduling on Bottom Layer

The thermal-aware task scheduling on bottom layer adopts a list-based scheduling algorithm. The algorithm keeps a list of tasks that are ready to be executed. Then the tasks in the ready list is mapped according to the descending order of the priority of each task in the application. The algorithm then uses a cost function that evaluates the thermal and performance cost of mapping the task to each processor on the bottom layer and algorithm maps the task to the processor with the lowest cost value. After the tasks in the ready list are all scheduled, the algorithm updates the ready list and repeats the above procedures until the scheduling of the application is finished. This list-based scheduling algorithm achieves two objectives of reduce execution time and balance power consumption through delicately defining the priority of tasks and the cost function of task mapping.

For a given task $G(V, E)$, Eq. (1) defines the priority of each task $v \in V$, which is denoted as $\Phi(v)$. Eq. (1) is a recursive definition, where V_{chd} is the set of children of task v; v' is a child of v; e' is the edge connecting v and v'; and $\bar{B}$ is the communication bandwidth of the 3D-NoC. Actually, $\Phi(v)$ stands for the total execution time along the longest path from task v to the end of application. This definition of task priority has been used in previous task graph scheduling works like [14] for the minimization of the execution time of the application.

$$\Phi(v) = \begin{cases} 0 & \text{if } V_{chd}(v) = \emptyset, \\ \max\{\Phi(v') + \bar{B} \times w(e')\} & \text{if } V_{chd}(v) \neq \emptyset. \end{cases} \quad (1)$$

The cost function for mapping each task is defined in Eq. (2). In the equation, $t_s(v, p)$ is the earliest starting time of task v on processor p, which is determined by the mapping of previous tasks; $Q(v, p)$ is the accumulated heat generated inside the processor, which evaluates the thermal impact of the previously mapped tasks on processor p; and α is a tuning parameter which is used to change the weight of the thermal related term in the cost function.

$$c(p, v) = t_s(v, p) + \alpha \times Q(v, p) \quad (2)$$

The definition of $Q(v, p)$ is further expanded in Eq. (3). Here v^* represents the task that has already been mapped to processor p; and q_{out} is an estimation of heat dissipated from the processor to the environment in unit time. The first two terms of Eq. (3) represent the total heat generated inside the processor while the last term evaluates the total amount of heat dissipation of the processor till the task finishes.

$$Q(v, p) = \sum_{v^*} \epsilon(v^*) + \epsilon(v) - q_{out} \times (t_s(v, p) + l(v)) \quad (3)$$

Since Eq. (2) evaluates both execution time and thermal effect, using it as the cost function in the mapping algorithm can address both objectives of reducing execution time and peak temperature of the system.

Algorithm 1 In-stack adjustment algorithm

Input: $G(V, E)$, M_b, P_b, P_t, Φ
Output: M

1: **for all** $p_b \in P_b$ **do**
2: V_s=find_tasks_mapped_in_stack(p, M_b)
3: V_s=sort_tasks_by_priority(V_s)
4: U_{pg}=divide_parallel_groups(V_s)
5: p_t=find_top_layer_processor(p)
6: $V_{ndef} = \emptyset$
7: **for all** $V_{pg} \in U_{pg}$ **do**
8: **if** $number\_of\_tasks(V_{pg}) > 1$ **then**
9: V_t=adjust_tasks_to_top_layer(V_{pg})
10: $V_b = V_s \cap \bar{V_t}$
11: **for all** $v \in V_{pg}$ **do**
12: $M(v) = (v \in V_b)\ ?\ p_b : p_t$
13: **end for**
14: **else**
15: V_{ndef}= $V_{ndef} \cup \{v\}$
16: **end if**
17: **end for**
18: **for all** $v \in V_{ndef}$ **do**
19: o_b=compute_overhead(v, p_b)
20: o_t=compute_overhead(v, p_t)
21: $M(v) = (o_b - o_t < o_{th})\ ?\ p_b : p_t$
22: **end for**
23: **end for**

B. In-Stack Adjustment to Top Layer

After scheduling algorithm on the bottom layer was completed, the in-stack adjustment algorithm will be performed. The algorithm exploits the parallelism of the tasks and selectively picks some of the tasks in each stack and moves them to the top layer processor in order to further reduce execution time using parallel processing while maintaining low temperature. Algorithm 1 shows the pseudo code of the in-stack adjustment algorithm.

In each stack, the set of tasks that have been mapped to the stack, V_s, is divided into multiple *parallel groups*. Here, U_{pg} is the set of all parallel groups, denoted as $\{V_{pg}\}$. As implied by the name, a parallel group, V_{pg} is a group of tasks that can be executed in parallel on different processors. The formation of parallel groups is performed as follows: firstly, pick a task with maximum priority from V_s as the head task of the parallel group, v_{head}; secondly, insert all tasks from V_s which fulfill the requirement in Equation (4) into the parallel group. Equation (4) guarantees that all the tasks inside V_{pg} do not have direct precedence relationship with each other, which makes the tasks in V_{pg} able to be executed in parallel. By repeating the two steps, V_s is eventually divided into multiple parallel groups.

$$\Phi(v_{head}) - l(v_{head}) > \Phi(v), v \in V_s \tag{4}$$

Because tasks of one parallel group can be executed in parallel, we can then move some of the tasks inside the parallel group to the top layer processor to reduce the execution time. This job is done by Line 7 to Line 17 of Algorithm 1. Note that only the parallel groups which contain more than one task can be adjusted (Line 8 to 16), and those parallel groups with single task are merged into set V_{ndef} (Line 15) for later consideration. Line 9 selects the tasks to be moved to the top layer processor, while the rest tasks of V_s will remain on the bottom layer processor (Line 10). The tasks of V_t and V_b must fulfill two conditions. Firstly, for any $v_t \in V_t$ and $v_b \in V_b$, there is $\rho(v_t) < \rho(v_b)$. It guarantees that the power consumption of tasks mapped to top layer are lower than tasks mapped to the bottom layer. It helps maintain low temperature of the system, as discussed in Section II. Secondly, there stands $\sum l(v_t) < \sum l(v_b)$. It further limits the total length of tasks mapped to the top layer to be smaller than tasks mapped to the bottom layer, which prevents temperature increase when there is load imbalance between the processors in one stack.

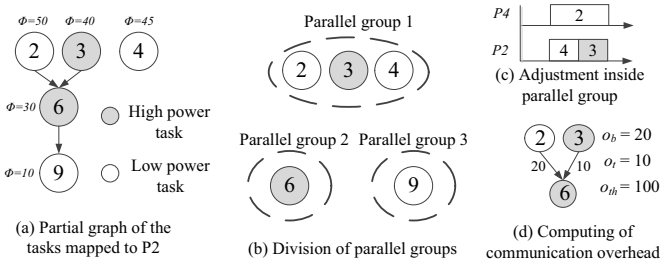

Fig. 4. In-stack adjustment algorithm example. (Inside the stack of processor P2 and P4 as shown in Fig. 3(b).)

Line 18 to 22 of Algorithm 1 deals with tasks from parallel groups with single task. The mapping of these tasks depends on the communication overhead between tasks as defined in Equation (5). The equation computes the total amount of messages on edges connecting task v and other tasks that are previously mapped to the processor on the other layer. As shown in Line 21 in Algorithm 1, only when the communication overhead of mapping task v to the bottom layer is too high, will it be mapped to the top layer. This is because mapping tasks to top layer processor has negative effects on reducing peak temperature of the system, and thus we only do this when the gain in execution time reduction is significant enough. The worst-case complexity of in-stack adjustment algorithm is $O(V)$.

$$\begin{cases} o_b = \sum_{v_t} w(e_t), e_t = v_t \rightarrow v, \text{or } v \rightarrow v_t, \\ o_t = \sum_{v_b} w(e_b), e_b = v_b \rightarrow v, \text{or } v \rightarrow v_b. \end{cases} \tag{5}$$

Fig. 4 illustrates the details of in-stack adjustment algorithm using the example shown in Fig. 3.

V. Experiment

A. Experiment Setup

The experiments are carried out on a 2-layer 8-core 3D-NoC system where each layer in the 3D-NoC contains four processing unit. In addition, we assumed that each processing unit in the target 3D-NoC system is a 32nm ARMv7 processor. HotSpot 5.02 [15] was used as the thermal simulation tool. The thermal parameters of HotSpot are shown in Table II. The other parameters followed the default settings of HotSpot.

In the experiments, we used two kinds of benchmarks: randomly generated task graphs and real application based task graphs. The randomly generated task graphs were selected from Standard Task Graph (STG) [16]. Five sets of randomly generated task graphs were used in the experiments; each set contained 100 task graphs; the number of nodes inside each task graph were 50, 100, 300, 500, 750

TABLE II
Experiments of real application based task graphs

Parameter	Value
Die thickness	$0.15mm$
Inter-layer material thickness	$0.02mm$
Die capacitance	$1.63 \times 10^6\ J/(m^3 \cdot K)$
Die resistance	$0.076mK/W$
Inter-layer material resistance	$0.25mK/W$
Environment temperature	$25°C$

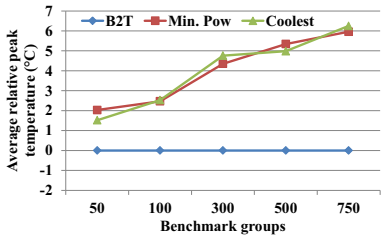

Fig. 5. Comparison of the relative peak temperature.

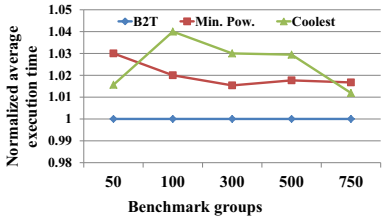

Fig. 6. Comparison of average normalized execution time.

respectively. The real application based task graphs were adopted from [17].

In order to verify the performance of B2T thermal-aware scheduling algorithm, we compared our algorithm for 3D-NoC with two dynamic thermal-aware task graph scheduling algorithms : minimum accumulated power (Min. Pow.) algorithm [10] and coolest first scheduling (Coolest) algorithm [11].

In the experiment, we first run the scheduling algorithms to produce the schedule of tasks. Then the power traces of accumulated power consumption of all processor during the task execution were generated according to the schedule. Finally, the temperature profile of the 3D-NoC was generated through HotSpot simulation.

B. Experiment Results

Fig. 5 compares the average relative peak temperature of the five benchmark groups under the three algorithms. For clearance, we set the lowest temperature value as zero and reduced other values accordingly, since the average peak temperature value of the five benchmark group have large difference. From the figure we can see that our B2T scheduling algorithm achieves significant peak temperature reduction compared to the other two algorithms. An average of $4.01°C$ reduction in average peak temperature is achieved. The maximum reduction in average peak temperature is $6.25°C$. This is because our algorithm keeps low power tasks on top layer processors and high power tasks on bottom layer processors which results in more efficient heat dissipation. Fig. 6 compares the normalized average execution time of random task graphs achieved by the three algorithms. The average execution time achieved by each scheduling algorithm is normalized to the largest average execution time appered in the experiments. The figure shows that our algorithm outperforms the other two algorithms by 2% on average and maximum 4%.

To further verify the performance of Bottom-to-Top scheme, we carried out a set of experiments on real application based task graphs. As mentioned in Section VA, three task graphs, Fpppp, Sparse and Robot were selected from [17]. Table III compares the execution time (measured in 10^2 cycles) and peak temperature (measured in $°C$) of the three algorithms. B2T algorithm shows significant temperature reduction in all cases. For each of the three applications, up to $6.48°C$, $6.52°C$ and $7.95°C$ reduction are respectively achieved. Although the execution time of the real applications resulted by our B2T scheme is not always the minimum, the increase in execution time within 5% is acceptable considering the significant amount of peak temperature reduction achieved.

TABLE III
EXPERIMENTS OF REAL APPLICATION BASED TASK GRAPHS

Benchmark	Robot		Sparse		fpppp	
Algorithm	Time	Temp.	Time	Temp.	Time	Temp.
B2T	1264	100.51	596	98.21	1350	104.18
Min. Pow.	1236	105.80	568	104.73	1284	112.13
Coolest	1243	107.09	613	103.97	1280	110.35

VI. CONCLUSION

In this paper, we propose an efficient thermal-aware task graph scheduling algorithm called Bottom-to-top scheme. The scheme first maps the task graph on the bottom layer of the 3D-NoC and then selectively adjusts some task to the top layer to reduce execution time while maintaining low operating temperature. Experimental results have shown up to $7.95°C$ reduction in peak temperature and up to 4% execution time reduction when compared to previous task graph scheduling algorithms designed for 2D-NoC systems.

ACKNOWLEDGEMENT

This work was supported by MoE AcRF Tier 2 grant (MOE2012-T2-1-126) of Singapore.

REFERENCES

[1] A. B. Ahmed and A. B. Abdallah, "Low-overhead routing algorithm for 3D Network-on-Chip" 2012, *ICNC*, pp. 23-32.

[2] B. Feero and P. Pande, "Networks-on-Chip in a three-dimensional environment: q performance evaluation," *IEEE Transanctions on Computers*, vol. 58, no. 1, pp. 32–45, 2009.

[3] D. Donald and M. Martonosi, "Techniques for multicore thermal management: classification and new exploration," 2006, *ISCA*, pp. 78–88.

[4] J. Hu and R. Marculescu, "Energy-aware communication and task scheduling for Network-on-Chip architectures under real-time constraints," 2004, *DATE*, pp. 234–239.

[5] X. Zhou, J. Yang, Y. Xu, Y. Zhang, and J. Zhao;, "Thermal-aware task scheduling for 3D multicore processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 1, pp. 60–71, 2010.

[6] J. Choi, C.Y.Cher, H. Franke, H. Hamann, A. Weger, and P. Bose, "Thermal-aware task scheduling at the system software level," 2007, *ISLPED*, pp. 213–218.

[7] A. Kumar, L. Shang, L. Peh, and N. Jha, "HybDTM: A coordinated hardware-software approach for dynamic thermal management," 2006, *DAC*, pp. 548–553.

[8] M. Chrobak, C. Dürr, M. Hurand, and J. Robert, "Algorithms for temperature-aware task scheduling in microprocessor systems," *Algorithmic Aspects in Information and Management*, pp. 120–130, Springer, 2008.

[9] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," 2007, *ICCAD*, pp. 281–288.

[10] Y. Xie, N. ViJ'aykrishnan, M. Kandemir, and M. Irwin, "Thermal-aware task allocation and scheduling for embedded systems," 2005, *DATE*, pp. 898-899.

[11] K. Stavrou and P. Trancoso, "Thermal-aware scheduling for future chip multiprocessors," *EURASIP Journal on Embedded Systems 2007*, pp. 40-40,2007.

[12] H. Wang, Y. Fu, and T. Liu, "Thermal management via task scheduling for 3D NoC based multi-processor," 2010, *ISOCC*, pp. 440–444.

[13] K. Puttaswamy and G. Loh, "Thermal analysis of a 3D die-stacked high-performance microprocessor," 2006, *GLSVLSI*, pp. 19–24.

[14] Y. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors" *ACM computing surveys*, vol. 31, no. 4, pp. 406–471, 2010.

[15] K. Skadron and et al., "Temperature-aware microarchitecture," 2003, *ISCA*, pp. 2–13.

[16] T. Tobita and H. Kasahara, "A standard task graph set for fair evaluation of multiprocessor scheduling algorithms," *Journal of Scheduling*, vol. 5, pp. 379–394, 2002.

[17] W. Liu and et al., "A NoC traffic suite based on real applications," 2011, *ISVLSI*, pp. 66–71.