An Efficient Task Scheduling Algorithm with Power-Aware Optimization for Network on Chip

(Full Paper: https://doi.org/10.1016/j.sysarc.2016.04.006)

Wei Hu[1,2,*], Qingsong Shi[3], Yonghao Wang[4], Kai Zhang[1,2], Jun Liu[1,2], Xiaoming Liu[1,2], Hong Guo[1,2]

[1]College of Computer Science and Technology,Wuhan University of Science and Technology, Wuhan, Hubei, China, 430065

[2]Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan, Hubei, China, 430065

[3]College of Computer Science, Zhejiang University, Hangzhou, Zhejiang, China, 310027

[4]Centre of Digital Media Technology, Faculty of Computing, Engineering and the Built Environment, Birmingham City University, United Kingdom

[*]Corresponding Author: huwei@wust.edu.cn

Abstract: More and more cores are integrated onto a single chip to improve the performance and reduce the power consumption of CPU without the increased frequency. The cores are connected by lines and organized as a network, which is called network on chip (NOC) as the promising paradigm of the processor design. However, it is still a challenge to enhance performance with lower power consumption. The core issue is how to schedule the tasks to the different cores to take full advantages of the on-chip network. In this paper, we proposed a novel scheduling algorithm with power-aware optimization for NOC. The traffic of the tasks will be analyzed. The tasks of the same program with high communication with the others will be mapped to the on-chip network as neighborhoods. And then the tasks of different programs are mapped to the cores step by step. The mapping of the tasks and the cores is computed at run-time dynamically and implement the online scheduling. The experimental results showed that this proposed algorithm can reduce the power consumption in communication with the performance enhanced.

Keywords: power consumption; scheduling algorithm; communication; network on chip

1. Introduction

With the development of semiconductor technology, more and more transistors can be integrated onto a single chip. However, it also has a big problem to increase the frequency of the single processor core for the faster growth of power-consuming[1]. The hardware manufacturers have begun to focus on the development of on-chip systems with more than one core. More processor cores are integrated onto the CPU, which is called Chip MultiProcessor(CMP). The multiple processor cores enhance the performance of the system without increasing the frequency of CPU. Bus-based communication is the traditional architecture to connect the on-chip devices, which is faster but needs more on-chip size. If more cores have the communication requirements, the bus itself will be the bottleneck of the CPU performance both in power-consuming and the transmission speed. It results in system performance degradation. Network on chip (NOC) is proposed as the promising diagram to solve this problem[2-4]. Typical NOC has an on-chip network to connect the processor cores, the processing units, memory blocks or the other on-chip devices[5]. The processor cores on NOC are distributed on the chip via the lines not the traditional buses[6-7]. The on-chip network makes NOC more scalable in communication. On-chip cores can communicate with each other through the network via on-chip routers for high efficiency. The design of NOC is communication-centric, not computation-centric[8]. When more and more processor cores are integrated onto a single chip, the communication will be more important than the computation.

The applications are divided into a plurality of parts in the system for running simultaneously on the multiple cores in NOC. The tasks from the partitioning is very important to improve the efficiency for the multi-core processors. They have to communication with each other according to their relationship to complete the target of the applications. The different applications have different traffic characteristics, which are the necessary requirements for NOC. When the traffic is heavy, the long latency from ends to ends will affect the system performance seriously and the power consumed by such traffic will be the main portion of the system power consumption[9]. When there are many tasks, the on-chip distribution of these tasks has great impact on the performance and power consumption. It plays an important role in performance enhancing and power saving to schedule the tasks to the cores for less traffic. In this paper, a novel online scheduling approach is proposed to provide the task scheduling with high efficiency for NOC. The communication between the tasks are analyzed. And the scheduling algorithm re-maps the tasks according to the analysis results.

This paper is organized as the follows. Section 2 describes the related works. Section 3 describes the system model of NOC. Section 4 presents the design of the algorithm. The experiments and results are described and discussed in Section 5. And at last, we give the conclusions and future work in Section 6.

2. Related Work

NOC has new hardware structure and it has caused the changes in the communication patterns. The communication is distributed on the network according to the requirements from the tasks. The structures of NOC itself are optimized to provide better support from the infrastructure of the on-chip network[10-13]. Compared with the traditional bus structures, the communication on lines will consume more time. The delay on lines may be the bottleneck of the performance. How to map the tasks to the cores is one of the key paths to solve the above problem. There are existing works on this problem to provide better solutions. The mapping algorithms are proposed first including the heuristic mapping algorithm[14], genetic algorithm based approaches[15], the algorithms with QoS[16] and Mesh oriented multi-target algorithms[17]. In such algorithms, the mapping process is determined before the operation of the system. It is an advantage that the mapping can be optimized and has good performance. However, such mapping cannot be adjusted according to the specific situations at run-time. As the changes occur, the mappings have to be re-computed. This is a very frustrating and time-consuming process.

Online scheduling is also an important research area. Run-time mapping is similar to the scheduling in traditional operating system. The tasks will be mapped dynamically according to the run-time environments. [18] focused on the resource allocation and thread immigration at run-time from the network characteristics of NOC. [19] proposed task predication and allocation based on the common task sequence constructed through the user habits as the reference impact. [20] assumed the cores had different power consumption level. And it proposed that the power consumption could be reduced by allocating tasks to the cores with lower power-consuming as much as possible. Scenario based mapping was provided in [21]. The scenarios were taken as state machine and the scenario transitions were the task scheduling. Common tasks were separated from the tasks in operating system in [22]. They were mapped to different cores independently to reduce the interferences of such tasks. DVS/DVFS was also used in task scheduling for power saving[23-25]. In such approaches, the frequency or voltage of the cores might be adjusted according to the run-time analysis. The frequency or voltage scaled down could reduce the power consumption of the whole system.

The tasks running on NOC have the communication requirements to achieve the target. The scheduling of the tasks on the network determines the traffic density. The related works shows that the scheduling algorithm was important for the performance of NOC. Optimized scheduling algorithm can also reduce the power consumption. In this paper, a dynamic online scheduling algorithm is proposed. It relies on the analysis of the traffic in the on-chip network at run-time. This algorithm can reduce the power consumption of NOC and have good impact on the performance.

3. System Model

NOC has new features and different designs. In this section, the system model is introduced including the on-chip network topology, the routing policy, task model and the energy model. They are the basis of the scheduling algorithm.

3.1 On-Chip Network Topology

NOC is a novel design diagram of system on chip (SOC), which has many advantages over bus-based communication. The on-chip network connects the on-chip devices and provides better performance. Various network topologies have been proposed for NOC architecture such as Ring, Mesh and Torus[26]. Mesh is the typical mainstream choice of NOC design. The mesh-based NOC architecture is shown in Fig.1.

Mesh has similar structure to matrix. The wires are used to connect the tiles as shown in Fig. 1 (a). Tiles are the nodes of the on-chip network. Each tile has the following components including routers, input/output interfaces, processor cores and on-chip memory(cache/SPM) as shown in Fig. 1 (b) as shown in Fig. 1 (a). The routers and input/output interfaces are responsible for the data forwarding and the communication between the cores. The tile may have non-conventional process core. There are probably processing elements (PE) for special purposes in tiles. Cache/SPM is local memory for the cores or PEs.

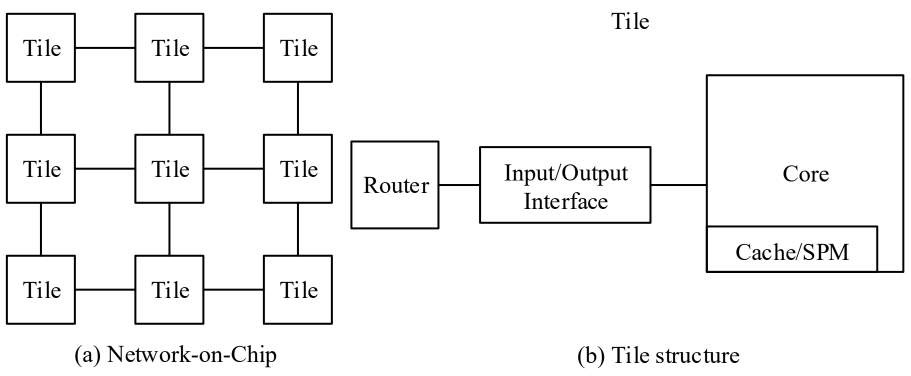


Fig. 1 Mesh Based Network on Chip

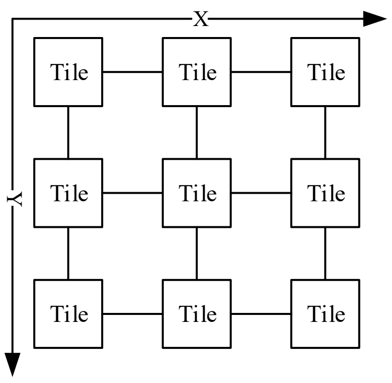


Fig. 2 X-Y Routing

3.2 Routing Policy

Routing policy provides the mechanism to determine the paths from source nodes to destinations. A routing algorithm is designed to improve the performance of on-chip communication and solve the problems such as deadlock and congestions. The routing algorithms can be classified into two types including deterministic routing and adaptive routing [8]. If a deterministic routing algorithm is used, the traversal paths are determined of all packets transmitted from the source node to destination node in the network. Adaptive routing chooses the routing direction according to the run-time environments dynamically. Such adaptive routing has strict requirements on router design which causes the complex router design.

X-Y routing is deterministic routing algorithm for mesh-based NOC[27]. In X-Y routing, the mesh structure is marked as X direction and Y direction. The routing is first along the X direction to forward the packets and then the Y direction as shown in Fig. 2. For a given destination, X-Y routing can reach it without deadlock. It has high simplicity and is easy to implement. In this paper, X-Y routing is chosen as the on-chip routing algorithm for the least impact by routing algorithm itself.

Switching mode is important for data transferring on NOC. Wormhole switching is packet based switching mode and it is one of the primary switching mechanisms currently [29]. The packets are splited into several small segments called flits. These flits are transferred through the network. If there are enough buffers for one flit, this flit can be buffered and forwarded by the router. Such method has reduced the network latency and saved the buffer. If the flits are blocked, the following flits can be forwarded by the other routers, which enhances the throughput of the network. In this paper, the wormhole switching is adopted as the switching mode. The disadvantage of wormhole routing is the latency may increase significantly when the traffic is heavy. Such situations are rare according to our setup and have few impact on the achievements.

3.3 Task Model

Application Control Graph (ACG) is used to present an application program, which is similar to the offline analysis in [16]. A typical ACG is shown in Fig. 3. An application program can be presented as G(V, R), in which V is the vertices set and R is the directed edge set between the vertices. Each V_i represents a task in program G. The tasks of G cannot be partitioned into smaller units. Each R_{ij} represents the connection between V_i and V_j. $F(R_{ij})$, which is the number of flits between V_i and V_j, is the weight for the traffic between the two tasks. CETA method[30] is used to obtain the ACG of an application program and SIMICS[31] is used as the platform to obtain the real-time traffic of the tasks inside the program.

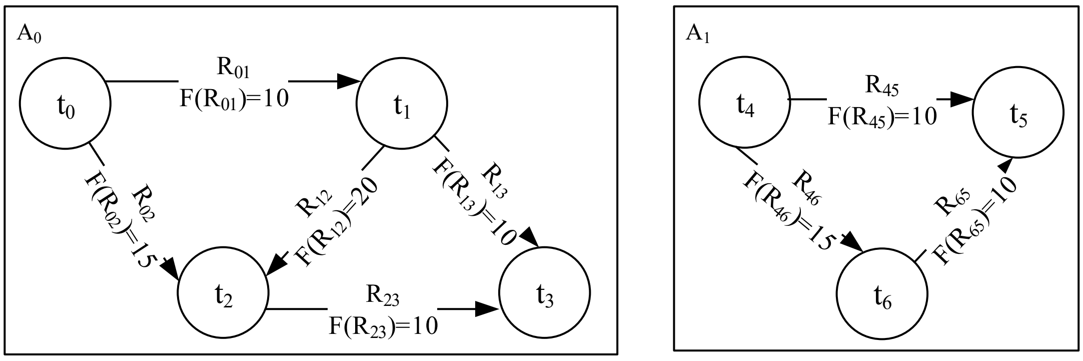


Fig. 3 An Example of ACGs

For all of the application programs in the system, there is a program set $A=\{A_0, A_1, A_2, \ldots, A_m\}$. Each A_i has different $Num(A_i)$ tasks. The system task model is represented as $T=\{T_0, T_1, T_2, \ldots, T_n\}$, where $Num(T) = \sum_{i=0}^{m} Num(A_i)$. T is naturally partitioned into different sub task sets according to each A_i in set A.

3.4 Energy Model

NOC has many processor cores or processing units on chip. And the tasks communication with each other through the on-chip lines. Cores/Pes have computation tasks and consume the energy. Such energy consumed is computation energy. When the flits are transferred via lines and forwarded by routers, they also consume energy. Such energy consumed is communication energy. This paper focuses on the communication energy.

All of the flit transferred in NOC consists of many digital bits, which is the basic unit transferred. Bit energy is defined as the energy consumed by one bit transferred in the network. For a given program, $E_b(R_{ij})$ is the energy consumed by one bit in communication of task T_i and T_j:

$$E_b\left(R_{ij}\right) = E_{Rb}(R_{ij}) + E_{Lb}(R_{ij}) \tag{1}$$

E_{Rb} is the energy consumed by the routers and E_{Lb} is the energy consumed by the lines for one bit. The basic topology of on-chip network is represented as a coordinate system as shown in Fig. 4 to obtain the detail of energy consumed. Each tile has its corresponding coordinates to represent its position. For example, the tile with core C_1 is represented as $C_1(1,0)$. For a given tile with core C_i, its coordinate is $C_i(X, Y)$. X_{Ci} and Y_{Ci} are horizontal coordinate and longitudinal coordinate respectively.

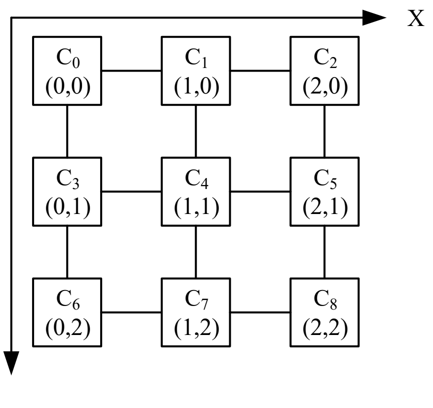


Fig. 4 On-Chip Network with Coordinates

And then Manhattan Distance is used to measure the distance of the cores/PEs:

$$\mathrm{D}(C_{ij}) = |(X_{Cj} - X_{Ci}) + (Y_{Cj} - X_{Ci})| \tag{2}$$

E_{link} is the energy consumed by one bit on one unit of Manhattan distance. E_{Router} is the energy consumed by one bit in a single router. Thus $E_{Rb}(R_{ij})$ can be obtained as the follows:

$$E_{Rb}(R_{ij}) = (D(C_{ij}) + 1) * E_{Router} \tag{3}$$

$E_{Lb}(R_{ij})$ is:

$$E_{Lb}(R_{ij}) = D(C_{ij}) * E_{Link} \tag{4}$$

According to (1), the total energy consumed by one bit is:

$$E_b(R_{ij}) = (D(C_{ij}) + 1) * E_{Router} + D(C_{ij}) * E_{Link} \tag{5}$$

Num(flit) is the number of data bits in one file. For one application program A_k, its total energy consumed $E(A_k)$ is:

$$E(A_K) = \sum_{\forall R_{ij} \in G \ of \ Q} F(R_{ij}) * Num(flit) * ((D(C_{ij}) + 1) * E_{Router} + D(C_{ij}) * E_{Link})$$

(6)

Set A={A_0, A_1, A_2,…, A_m} has all of the active programs of the system, during the given time slot S(0, t). And then the total energy consumed by the system is:

$$E_A = \sum_{i=0}^{m} E(A_i) \tag{7}$$

(6) and (7) shows that the following factors have impacts on the total energy including the number of flits, the Manhattan distance, the energy consumed by router and line. The number of flits is determined by the communication requirements and the retransmission mechanism. The energy consumed by router and line is assumed as constants in this paper. Thus how to reduce the distance of the tasks is the main focus.

4. Algorithm Design

In this section, the details of the scheduling algorithm are discussed. Firstly, the algorithm for a single program is presented, which maps tasks of the same program to the computational units on chip. And then the optimized algorithm for multiple programs is described. At last the online scheduling is discussed.

4.1 Algorithm Design for Single Program

The application program has more than one task partitioned in current system. The multiple tasks can provide higher parallelism on NOC than traditional single-core chip. These tasks should also be assigned to the cores on chip for the performance. One of the simplest scheduling algorithm for single program is X-Y first fit mapping(XYFF) for the program A_k, which is similar to the X-Y routing. When there is a task unmapped, this algorithm looks for an idle core in X direction first and then in Y direction if there is no suitable core in X direction. The mapping starts at $C_0(0, 0)$. This algorithm is simple and easy to implement. However the disadvantage is also obvious for the larger amount of communication.

As an optimization, the tasks can be mapped to the nearest cores according to their Manhattan distance. This is the nearest fit mapping (NF). There is no determined direction in this algorithm. The first task is still mapped to $C_0(0, 0)$. The next task will be mapped to the core with the smallest Manhattan distance with $C_0(0, 0)$. The following tasks are mapped to the cores with the smallest Manhattan distance with the previous mapped core. This algorithm will map the tasks to the cores as a cluster. It can reduce the traffic by shortening the total Manhattan distance. However, NF still has its disadvantage. The mapping in NF can achieve the optimization in the nearest two tasks for each mapping. But it cannot map the tasks to reduce the total communication energy.

The total Manhattan distance should be reduced further. Energy saving mapping(ES) aims to achieve this target. The tasks with the heaviest traffic should be mapped nearly according to the analysis in energy model. The traffic of a task can be obtained as the follows:

$$Trafiic(T_i) = \sum_{\forall R_{ij} \in A_k} F(R_{ij}) \quad (8)$$

The task with heaviest traffic is mapped first to $C_0(0, 0)$. The next task to be mapped is the task with heavy traffic with the previous being mapped task on $C_0(0, 0)$ and this task is mapped to the core with smallest Manhattan distance to $C_0(0, 0)$. The candidate cores are $C_1(0, 1)$ and $C_3(1, 0)$. After mapping, the following tasks are mapped according to the same method until all the tasks are mapped to the cores. However, $C_0(0, 0)$ is not the best start point. If the task with heaviest traffic communicates with more than two tasks, the Manhattan Distance of the extra tasks and start task increases according to the mapping mechanism. The improvement can be got through the following method. Assuming $Num(A_k)$ is the number of tasks in A_k, the center of a region which has $Num(A_k)$ cores is found to use as the start point of the mapping.

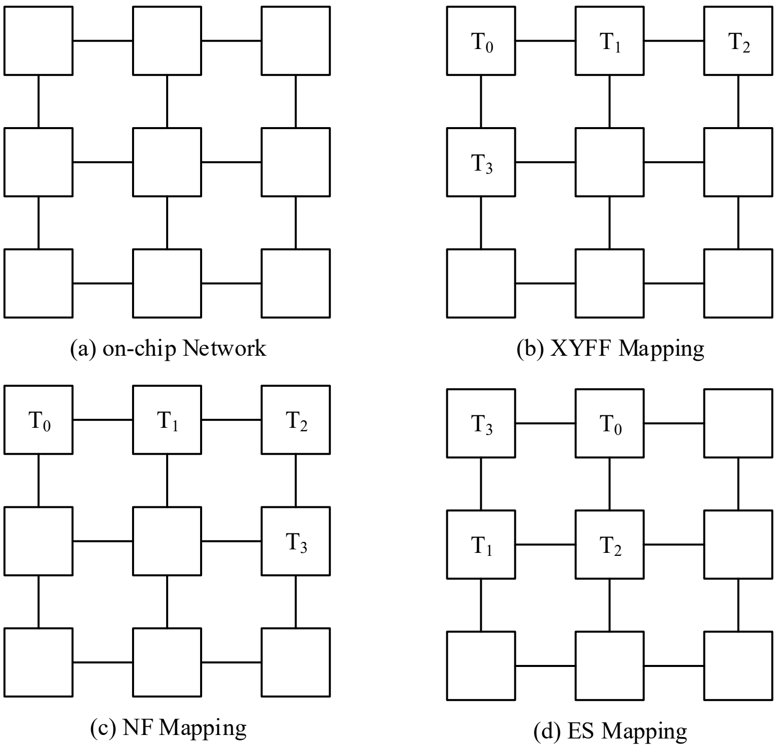


Fig. 5 An Example: Algorithm Design for Single Program

Fig. 5 shows an example for the above algorithms by using the tasks of A_0 shown in Fig. 3. XYFF, NF and ES mapping have maximum hops for three, three and two respectively. The average hops of the three algorithms are 1.67, 1.67 and 1.33. The total energy consumed by the three algorithms is shown in the following.

$$E_{XYFF} = 175Num(flit) * E_{Router} + 110Num(flit) * E_{Link}$$
$$E_{NF} = 155Num(flit) * E_{Router} + 90Num(flit) * E_{Link}$$
$$E_{ES} = 150Num(flit) * E_{Router} + 85Num(flit) * E_{Link}$$

Though the maximum hops and average hops of NF are same to XYFF, the shortened Manhattan Distance reduces the energy consumed. ES has the best performance. It has reduced the maximum hops and average hops. This makes ES be able to enhance the communication efficiency. And ES has consumed fewer energy according the above computation.

4.2 Algorithm Design for Multiple Programs

When there are more programs in the system, more tasks should be mapped to the network. As the analysis in Section 4.1, ES will gather the tasks of the same program as a cluster. ES can be used as the basis for the scheduling of multiple tasks. As mentioned in Section 3.3, the task set T={T_0, T_1, T_2, …, T_n}, where $n = \sum_{i=0}^{m} Num(A_i)$. Each A_i has corresponding tasks and constructs a natural task set partition as represented as T(A_i).

For on-chip network N, it is presented as N (C, P). C is the set of processor core and C={C_0, C_1, C_2, …, C_p}. Num(C) is the number of all the cores in C. P is the set of paths P_{ij}, in which P_{ij} represents one path from the processor core C_i to the processor core C_j.

$$s = |C_i \to C_j| \tag{9}$$

s means that the number of routers on NOC from the processor core C_i to the processor core C_j.

h(C_i) represents the number of the processor cores in all directions which are directly connected to the processor core C_i; C(C_i) represents the set of the processor cores which are directly connected to the processor core C_i.

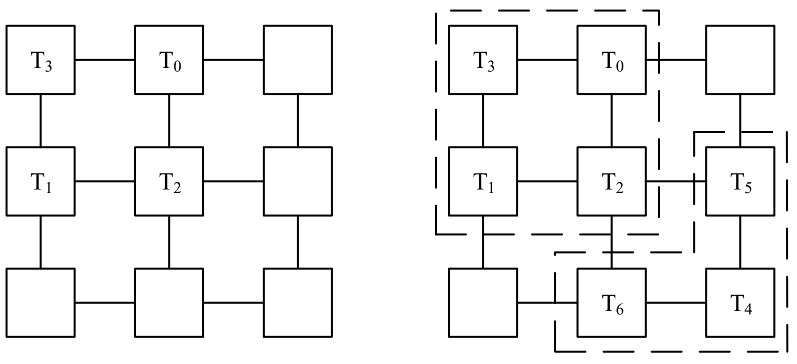
The partitioning of the on-chip network is based on the number of the programs. If $\mathrm{Num(C)} \geq \sum_{i=0}^{m} Num(A_i)$, it means all of the tasks in set T can be assigned to the network. The partitioning can start at once. If $\mathrm{Num(C)} < \sum_{i=0}^{m} Num(A_i)$, it means there are not enough cores for all of the tasks. Before the partitioning, some tasks should be removed from the task set T' to be scheduled. First, all of the tasks after T_p in T is removed from the task set T'. Second, the program A_k is found, which contains task T_p. And then A_k is checked to find out whether task T_p is the last task in A_k. If it is, can be assigned to the network. Otherwise, all the tasks in A_k are removed from the task set T'. And A_{k+1} will be checked. If $\mathrm{Num}(A_{k+1}) < |\mathrm{Num(C)} - \mathrm{Num(T')}|$, the tasks in A_{k+1} is moved to T'. Otherwise the next program will be checked until all of the programs are checked. If $\forall A_i \notin T', \mathrm{Num}(A_i) < |\mathrm{Num(C)} - \mathrm{Num(T')}|$, T' is stable for the scheduling. For the simplicity of the algorithm description, it is assumed $\mathrm{Num(C)} \geq \sum_{i=0}^{m} Num(A_i)$.

The enhanced ES algorithm for multiple programs (MES) is designed to map the tasks in T' to the cores in C. Our method is using ES described in Section 4.1 for all the tasks. Each A_i is mapped to the cores via ES algorithm. When the tasks in A_0 is mapped, the on-chip network will be a non-standard mesh based network. C(C_i) is modified for the mapped cores. The corresponding cores are removed from the direct connection sets. Thus the mapped cores will not be mapped again. Such method can improve the map of the tasks according to the traffic analysis of the tasks in each A_i. The on-chip network is partitioned into different regions as a region set D={D_0, D_1, D_2, …, D_s}. Num(D) is the number of regions. Num(D_i) is the number of cores in region D_i. The partitioned regions are satisfied to the following two conditions:

$$\sum_{i=0}^{s} Num(D_i) \geq \mathrm{Num}(T') \tag{10}$$

$$Num(D_i) = Num(A_i),\ \forall A_i \subseteq T' \tag{11}$$

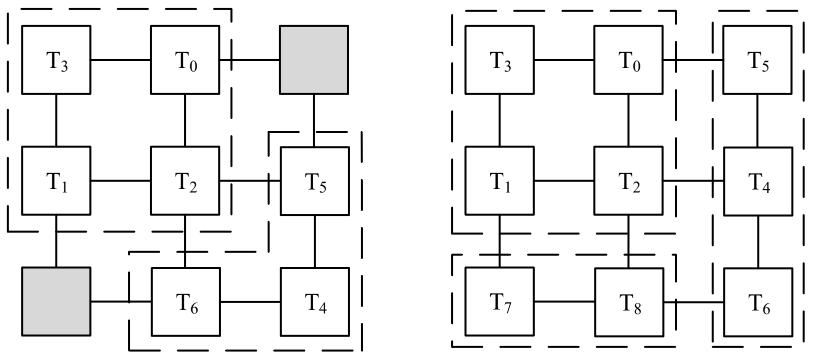
Fig. 6 shows an example for MES algorithm by using the tasks of A_0 and A_1 shown in Fig. 3. The tasks in A_0 is mapped firstly. T_2 has the heaviest traffic and it is mapped first. The other three tasks are mapped in order. The mapping result of A_0 is shown in Fig. 6(a). The tasks in A_1 is mapped to the cores next. T_4 has the heaviest traffic and it is mapped first. And then the other two tasks are mapped too. The result of the mapping is shown in Fig. 6(b).



(a) Mapping of A_0 (b) Mapping of A_1

Fig. 6 An Example: Algorithm Design for Multiple Programs using MES

The mapping can be finished from another point of view. The on-chip network is partitioned into different regions. Each A_i is mapped to the corresponding region D_i. Each region D_i has a task set A_i. The tasks from the same A_i will be mapped by using ES algorithm. This method (called ESR) is simple and easy to implement. All the tasks can be mapped to some core in C. The difference between ES and ESR is that more cores may be unmapped ones in ES and ESR has the possibility to contain more tasks. For example, if there are only A_0 and A_1 in A as shown in Fig. 3, ES and ESR have the same mapping results. However, if there is another A_2 which has two tasks T_7 and T_8, these extra tasks can be mapped to the cores using ESR. But the separate two cores in ES as shown in Fig. 7(a) are isolated from each other and they cannot be allocated. When ES is used to schedule the tasks, the energy saving is the first factor. ES can be seen as energy saving first algorithm. On the contrary, how to map all of the tasks is the first factor. Thus ESR can be seen as the task mapping first algorithm.



(a) Mapping by ES (b) Mapping by ESR

Fig. 7 Comparison of Mapping by ES and ESR

4.3 Online Scheduling

The scheduling described above is based on the traffic on chip known in advance. However, the exact communication overhead can only be obtained at run-time in a system. Though the traffic on chip can be profiled through the static analysis, the profiling information cannot guarantee the correct and the offline scheduling has no scalability when there are more available resources. The online scheduling is based on the profiling of the tasks at run-time. The tasks in the same region will be re-mapped if major changes occurred in the distribution of traffic. The online scheduling (OES) is shown in Fig. 8.

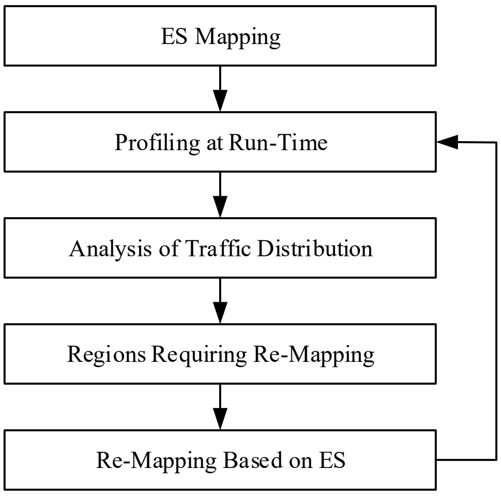

ES Mapping
Profiling at Run-Time
Analysis of Traffic Distribution
Regions Requiring Re-Mapping
Re-Mapping Based on ES

Fig. 8 Online Mapping Methodology

The traffic of the tasks is analyzed through the static profiling. This is the starting point of the scheduling. The tasks are mapped to the cores according to the ES method first. The on-chip network is partitioned into different regions. Each region has some tasks. The profiling information is collected at run-time. The traffic distribution is analyzed through CETA analysis. If major changes occur inside a region, the task in this region will be re-mapped according to ES.

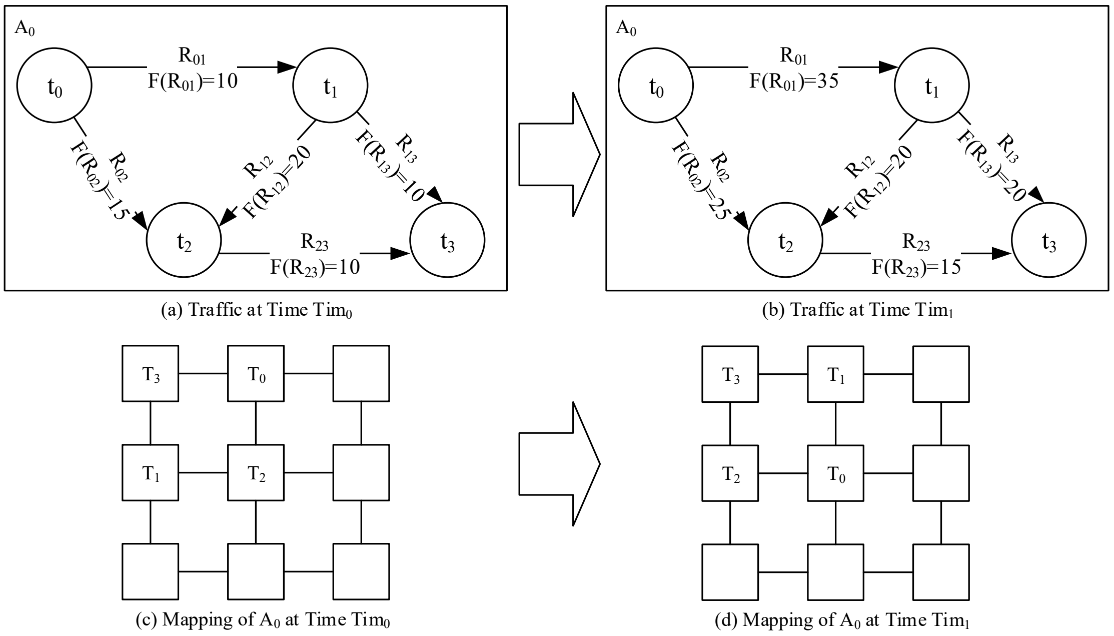


Fig. 9 Online Scheduling of A_0

For the given task set A_0, the traffic at time Tim_0 is shown in Fig. 9 (a). Task t_2 is the center of the traffic and mapped first at time Tim_0. The mapping result is shown in Fig. 9 (c). The traffic at time Tim_1 is shown in Fig. 9 (b). The traffic distribution has changed after execution from Tim_0 to Tim_1. Now task t_0 is the center of the traffic and this task becomes the starting point of the re-mapping. And then all the tasks in the same region are re-mapped according to ES method. The re-mapping result is shown in Fig. 9 (d). E_{ES} and E'_{ES} are the energy consumed without re-mapping and after re-mapping as shown in the follows:

$$E_{ES} = 265Num(flit) * E_{Router} + 150Num(flit) * E_{Link}$$
$$E'_{ES} = 250Num(flit) * E_{Router} + 135Num(flit) * E_{Link}$$

Online scheduling has reduced the energy consumption for the mapping of A_0 as shown above for the re-mapping cuts down the hops of the total communication. The reduced hops can also improve the performance of the systems.

5. Experimental Results and Analysis

5.1 Experiment Setup

The proposed algorithm has been tested based on the simulation. SIMICS is used to obtain the traffic of the tasks. They are used as the injection data to the on-chip network. The settings of SIMICS is shown in Table 1.

Table 1. SIMICS Configuration

Parameter	Configuration
CPU Number	16
Freq_mhz	60
Cache Size	32KB
Disk Size	4 GB
Memory Size	512 MB
Operating System	Linux 2.6.20

Noxim[33] is used as the basic NOC simulator. This platform provides the communication status on chip, the profiling information of the system performance and the energy consumed. The settings of Noxim is shown in Table 2.

Table 2. Parameter Settings of Noxim

Parameter	Settings
-mesh_dim_x	4 nodes
-mesh_dim_y	4 nodes
-buffer_depth	4
-max_packet_size	10 flits
-routing_algorithm	X-Y routing
-stats_warm_up_time	1000 cycles

The benchmarks are the following programs including Barnes, FFM_3, MPGdec, MPGenc, ocean_contiguous_partitions(Ocean), and Water_spatial(Waters). The traffic of these programs is obtained using CETA. The original cycles of the program are too large for the analysis and ECycle is used to record the cycles. One ECycle equals to 10000 original cycles. The basic parameters of these programs are shown in Table 3.

Table 3. Basic Parameters of Benchmarks

Program	Task Number	ECycles
Barnes	10	181
FFM	5	18
MPGdec	16	23
MPGenc	6	46
Ocean	6	137
Waters	9	63

Noxim is used to simulate the on-chip network. The computation of the tasks cannot be simulated for the cores cannot be simulated in Noxim. The communication overhead of the task migration is taken into account. ES algorithm has low complexity with $O(\mathrm{Num}(C) *$

$\mathrm{LogNum(C) + Num(T) * LogNum(T))}$. The time cost of task migration does not need to be considered as the main impact.

5.2 Results and Analysis

XYFF, NF, ES and OES are implemented and used in the experiments. Each program has different run-time features including the run-time traffic. The experimental results are normalized for the analysis with XYFF as the baseline.

Fig. 10 shows the average traffic on chip by counting the flits flowing through the routers. ES and OES can reduce the amount of flits transmitted through the on-chip network. The tasks of each program are gathered in one region according to their communication relationship. The tasks with intensive communication are mapped as neighbors as much as possible. The flits will pass through fewer routers and this can reduce the total amount of on-chip traffic. The proposed algorithm can reduce 57% (ES) and 64% (OES) of the total flits. This is the basis of the optimization for the energy saving and performance improvement.

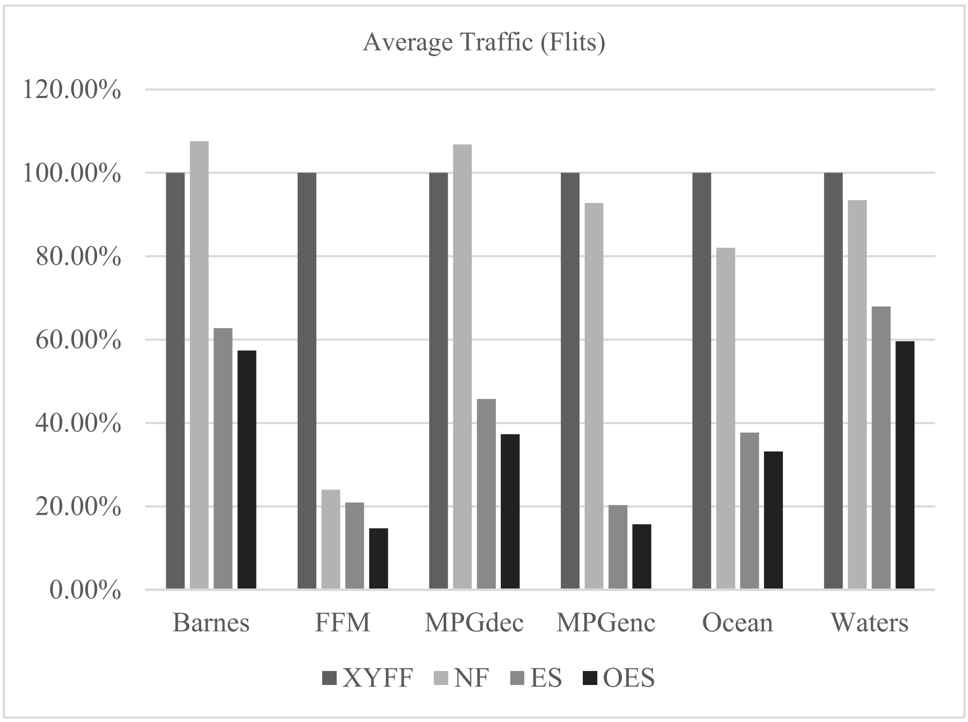


Fig. 10 Average Traffic

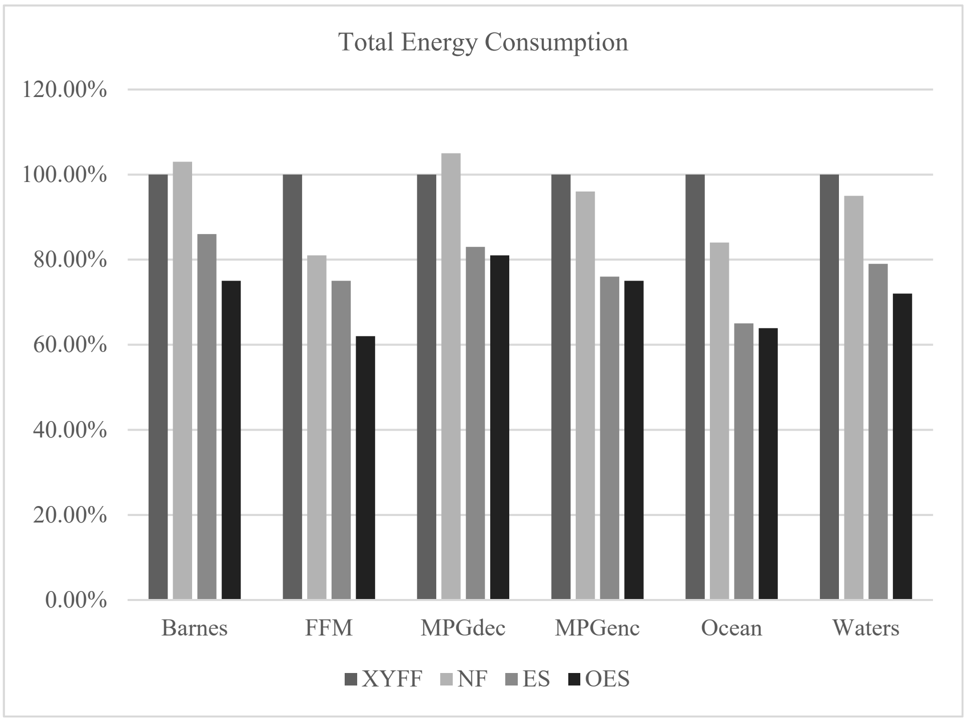


Fig. 11 Total Energy Consumption

The energy consumption is shown in Fig. 11. As shown in (6) and (7), the reduced traffic can save more energy. The total energy consumption is basically consistent with the average traffic. As Fig. 11 shows, OES cannot provide significant improvement by OES compared with ES. The possible reason is that MPGenc and Ocean has small number of tasks and these tasks have the balanced communication. The re-mapping by EOS cannot reduce more transmitted flits. And it results in the very limited improvement in energy consumption.

OES can re-map the tasks according to the profiling information at run-time. It provides more flexibility for the system as online scheduling. However, online scheduling needs to gather the information and re-map the tasks. Such operations will bring some extra loss for the system. The overhead of OES is shown in Fig. 12. The maximum overhead is 2.4% and the average overhead is 1.67%. The overhead is acceptable.

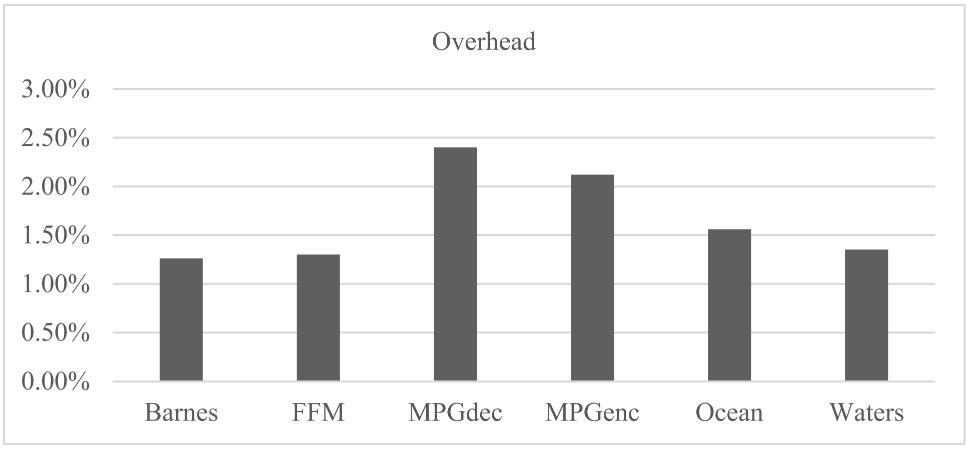


Fig. 12 Overhead of OES

Fig. 13 and Fig. 14 show the maximum latency and average latency of different algorithms respectively. The maximum latency may not be reduced according to Fig. 13. The reduced total flits do not mean that the maximum latency will be reduced. ES and OES may map the tasks without intensive communication to the cores with long Manhattan Distance. The efficiency of tasks with intensive communication is guaranteed first. Thus such tasks can communication with each other with fewer hops in ES and OES. The average latency is reduced about 22% and 29% in ES and OES respectively as shown in Fig. 14.

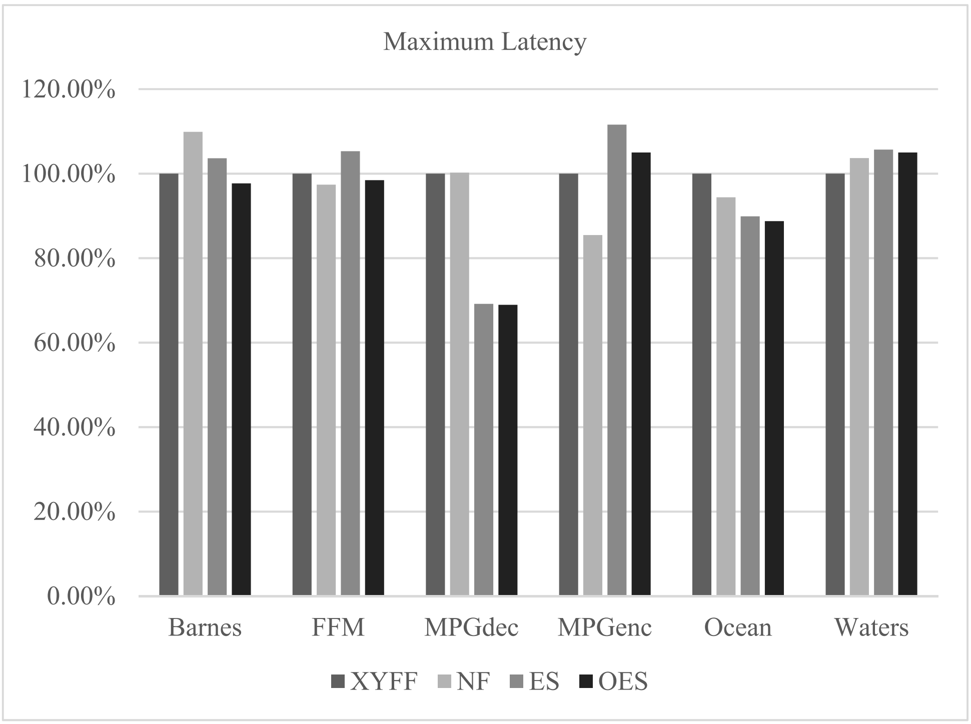


Fig. 13 Maximum Latency

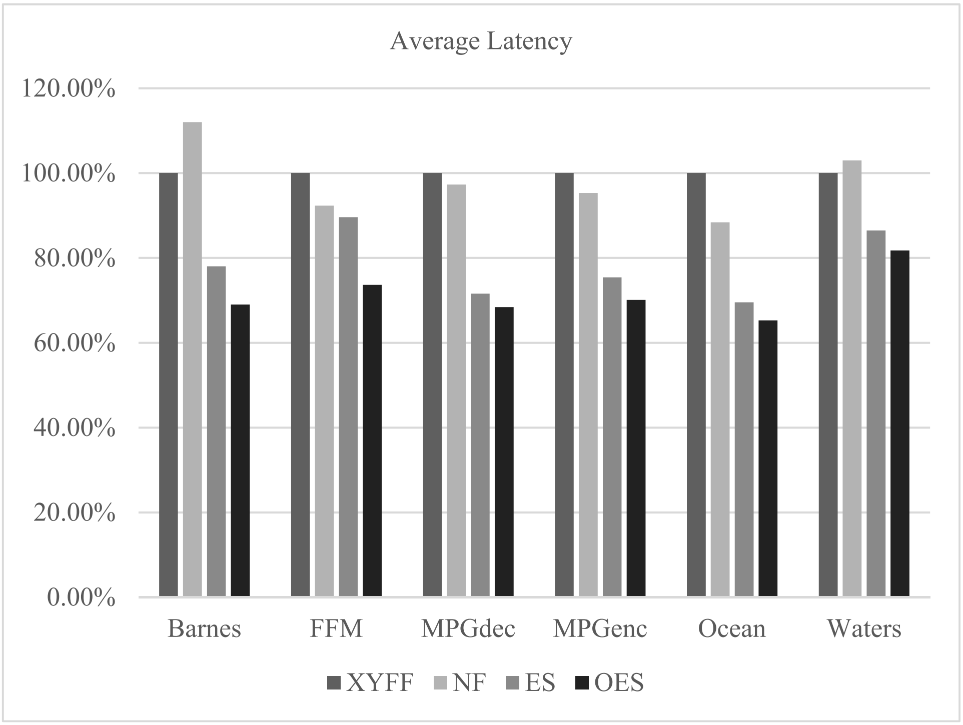


Fig. 14 Average Latency

According to the experimental results, OES is the best optimization algorithm for online scheduling for NOC based many-core system. This algorithm can schedule the tasks without obtaining the traffic in advance. Though there is also some overhead, OES can improve the performance and reduce the power consumption.

6. Conclusions and Future Work

More transistors are integrated onto a sing die for the advances in semiconductor technology. NOC is proposed as a promising diagram to break through the bottleneck of on-chip communication. The processor cores are connected by lines and organized as an on-chip network. The network traffic brings a new challenge to NOC in both the performance and the energy consumption. The communication characteristics of the different programs have great impact on the on-chip network. The key issue is how to map the tasks of the programs to the network. In this article, an optimized scheduling algorithm is proposed for NOC-based many-core system. The basic idea is that communication is the center of NOC architecture. An energy saving scheduling algorithm is proposed first to map the tasks to the cores according to the communication-intensive by using static analysis of the on-chip traffic. And then this algorithm is extended to online one based on the profiling information obtained at run-time. It re-maps the tasks for better performance and energy saving. The experimental results show that online analysis and remapping can save 29% energy in average with reduced latency compared with the off-line analysis.

More work needs to be done in the future. The algorithm proposed in this paper does not take the migration cost as account. When there are more tasks in the system, the migration may occur frequent. The cost will increase. Secondly, the global re-mapping should be taken as a design factor for the further improvement. And at last, this algorithm should be extended for the heterogeneous NoC.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Granted No. 61100055 and 31201121).

References

[1] M.S. Khaira. Micro-2010: lead performance microprocessor of the year 2010-myth or reality. In Proc. 12th International Conference on VLSI Design (VLSID99), 1999, 157-163.

[2] L.Benini and G.D.Micheli. Networks on chips:A New SoC Paradigm. IEEE Computer, 2002, 70-78.

[3] A. Ivanov and G.D.Micheli. The Network-on-Chip Paradigm in Practice and Research. IEEE Design and Test of Computer, 2005, 399-403.

[4] B. Vermeulen, J. Dielissen, K. Goossens and C. Ciordas. Bringing Communication Networks on a Chip: Test and Verification Implications. IEEE Commun. Magazine, 2003, 74-81.

[5] T. Mak, P.Y. Cheung, W. Luk and K. Lam, “A DP-network for optimal dynamic routing in network-on-chip”, Proc. the 7th IEEE/ACM international Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 09), ACM, New York, NY, Oct. 2009, pp. 119-128.

[6] C. Liu, E. Cota, H. Sharif and D. Pradhan. Test Scheduling for Network-on-Chip with BIST

and Precedence Constraints. In Proc. ITC, 2004, 1369-1378.

[7] M.B. Taylor and W. Lee. Scalar Operand Networks. IEEE Transactions on Parallel and Distributed Systems, Feb. 2005, vol.16, no. 2, 145-162.

[8] T. Bjerregaard and S. Mahadevan, “A survey of research and practices of Network-on-chip”, ACM Comput. Surv., vol. 38, issue 1, Jun. 2006, Article No. 1.

[9] K. Chang, J. Shen and T. Chen, “Tailoring circuit-switched network-on-chip to application-specific system-on-chip by two optimization schemes”, ACM Trans. Des. Autom. Electron. Syst., vol, 13, issue 1 , Jan. 2008, pp. 1-31.

[10] K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson and K. Chung. The Case for a Single-Chip Multiprocessor. ACM SIGOPS Operating Systems Review, vol 30, issue 5, 1996, 2-11.

[11] J. Meindl. Gigascale Integration: Is the Sky the Limit?. IEEE Circuits and Devices Maganize, vol 12, issue 6, 1996, 19-24.

[12] W.J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In Proc. the 38th Conference on Design Automation (DAC 01), 2001, 684-689.

[13] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini and G. De Micheli. NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip. IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 2, 2011, 113-129.

[14] W. H. Ho and T. M. Pinkston. A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns. In Proc. the 9th Intl. Symp. on High-Performance Computer Architecture, 2003, 377.

[15] K. Srinivasan and K.S. Chatha. ISIS: a genetic algorithm based technique for custom on-chip interconnection network synthesis. Proc. 18th Intl. Conf. on VLSI Design, 2005, 623-628.

[16] S. Murali, L. Benini and G. de Micheli. Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees. In Proc. Conf. on 2005 Asia and South Pacific Design Automation Conf., 2005, 27-32.

[17] G. Ascia, V. Catania, M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In Proc. 2004 Intl. Conf. Hardware/Software Codesign and System Synthesis, 2004, 182-187.

[18] D. Barcelos, E. W. Brião, F. R. Wagner. “A hybrid memory organization to enhance task migration and dynamic task allocation in NoC-based MPSoCs”, Proc. of the 20th annual conference on Integrated circuits and systems design, pp.282-287, 2007.

[19] C. Chou, R. Marculescu. “User-aware dynamic task allocation in networks-on-chip”, Proc. of the Conf. on Design, automation and test in Europe, pp.1232-1237, 2008.

[20] J. Hu and R. Marculescu, “Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints”, Proc. of the Conf. on Design, automation and test in Europe - Volume 1, pp.10234, 2004.

[21] L. Schor, I. Bacivarov and D. Rai et al. “Scenario-based design flow for mapping streaming applications onto on-chip many-core systems”, Proc. 2012 Intl. Conf. Compilers, architectures and synthesis for embedded systems (CASES '12). ACM, New York, NY, USA, pp. 71-80.

[22] P.C. Hsiu, D.N. Lee and T.W. Kuo. “Task synchronization and allocation for many-core real-time systems”. Proc. ninth ACM Intl. Conf. on Embedded software (EMSOFT '11). ACM, New York, NY, USA, pp. 79-88.

[23] . Wang, H. Liu and Z.W. Qin et al. “Overhead-aware energy optimization for real-time streaming applications on multiprocessor System-on-Chip”, ACM Trans. Des. Autom. Electron. Syst. Vol. 16, issue 2, Article 14, 32 pages, April 2011.

[24] D.S. Zhang, D.K. Guo and F.Y. Chen et al. “TL-plane-based multi-core energy-efficient real-time scheduling algorithm for sporadic tasks”, ACM Trans. Archit. Code Optim. Vol. 8, issue 4, Article 47, 20 pages, January 2012.

[25] C. Liu, J. Li and J. Rubio et al. “Power-efficient time-sensitive mapping in heterogeneous systems”, Proc. the 21st Intl. Conf. on Parallel architectures and compilation techniques (PACT '12), ACM, New York, NY, USA, pp. 23-32.

[26] H. Wang, L. Peh and S. Malik, “A Technology-Aware and Energy-Oriented Topology Exploration for On-Chip Networks”, Proc. the Conference on Design, Automation and Test in Europe (DATE 05), Mar. 2005, vol.2, pp. 1238-1243, doi: 10.1109/DATE.2005.40.

[27] J. Wu, “A deterministic fault-tolerant and deadlock-free routing protocol in 2-D meshes based on odd-even turn model”, Proce. the 16th international Conference on Supercomputing (ICS 02), Jun. 2002, pp. 67-76, doi: 10.1145/514191.514204.

[28] M. Pastrnak, P.H.N. de With, S. Stuijk and J. van Meerbergen, “Parallel implementation of arbitrary-shaped MPEG-4 decoder for multiprocessor Systems”, Proc. Visual Comm. and Image Processing (PWSM 06), San Jose, CA, Jan. 2006.

[29] S. Taktak, J. Desbarbieux and E. Encrenaz, “A tool for automatic detection of deadlock in wormhole networks on chip”, ACM Trans. Des. Autom. Electron. Syst., vol. 13, issue 1, Jan. 2008, pp. 1-22, doi: 10.1145/1297666.1297672.

[30] A.H. Liu and R.p. Dick, “Automatic run-time extraction of communication graphs from multithreaded applications”, Proc. the 4th IEEE/ACM international Conference on Hardware/Software Codesign and System Synthesis (ODES+ISSS 06), Oct. 2006, pp. 46 – 51.

[31] Virtutech Simics, http://www.virtutech.com/products/.

[32] T.T. Ye, G.D. Micheli and L. Benini, “Analysis of power consumption on switch fabrics in network routers”, Proc. the 39th Annual Design Automation Conference (DAC 02), Jun. 2002, pp. 524-529, doi: 10.1145/513918.514051.

[33] Noxim: Network-on-Chip simulator, available: http://sourceforge. net/projects/noxim