PAPER *Special Section on Solid-State Circuit Design-Architecture, Circuit, Device and Design Methodology*

Application-Specific Network-on-Chip Synthesis: Cluster Generation and Network Component Insertion*

Wei ZHONG[†a)]**,** ***Nonmember,*** **Yoshimura TAKESHI**[†]**,** ***Fellow,*** **Bei YU**[††]**,** ***Nonmember,*** **Song CHEN**[†]**,**
Sheqin DONG[†††]**,** ***Members, and*** **Satoshi GOTO**[†]**,** ***Fellow***

SUMMARY Network-on-Chips (NoCs) have emerged as a paradigm for designing scalable communication architecture for System-on-Chips (SoCs). In NoC, one of the key challenges is to design the most power-performance efficient NoC topology that satisfies the application characteristics. In this paper, we present a three-stage synthesis approach to solve this problem. First, we propose an algorithm [floorplanning integrated with cluster generation (FCG)] to explore optimal clustering of cores during floorplanning with minimized link and switch power consumption. Then, based on the size of applications, an Integer Linear Programming (ILP) and a heuristic method (H) are also proposed to place switches and network interfaces on the floorplan. Finally, a power and timing aware path allocation algorithm (PA) is carried out to determine the connectivity across different switches. Experimental results show that, compared with the latest work, for small applications, the NoC topology synthesized by FIP (FCG+ILP+PA) method can save 27.54% of power, 4% of hop-count and 66% of running time on average. And for large applications, FHP (FCG+H+PA) synthesis method can even save 31.77% of power, 29% of hop-count and 94.18% of running time on average.

key words: *networks on chips, floorplanning, topology synthesis*

1. Introduction

Network-on-Chips (NoCs) have been proposed as a solution for addressing the global communication challenges in System-on-Chip architectures that are implemented in nanoscale technologies [1] [2]. In NoCs, the communication among various cores is achieved by on-chip micro-networks components (such as switches and network interfaces) instead of the traditional non-scalable buses. Comparing with bus-based architectures, NoCs have better modularity and design predictability. Besides, the NoC approach offers lower power consumption and greater scalability.

NoCs can be utilized as regular or application-specific network topologies, as shown in Fig.1. For regular NoC topology design, some existing NoC solutions assume a mesh based NoC architecture [3] [4], and their focus is on the mapping problem. Regular NoC architectures offer lower design time, and are useful when implemented in a generic multiprocessor environment such as the MIT RAW [5]. On the other hand, for application-specific NoC topology design, the design challenges are different in terms of irregular core sizes, various core locations, and different communication flow requirements [6] [7] [8] [9]. Most SoCs are typically composed of heterogeneous cores and the core sizes are highly non-uniform. The application-specific NoC architecture with structured wiring, which satisfies the design objectives and constraints, is more appropriate. The application-specific NoC architecture has been demonstrated to be superior to regular architectures in terms of power, area and performance [10]. This paper concentrates on the synthesis method of application-specific NoC topologies.

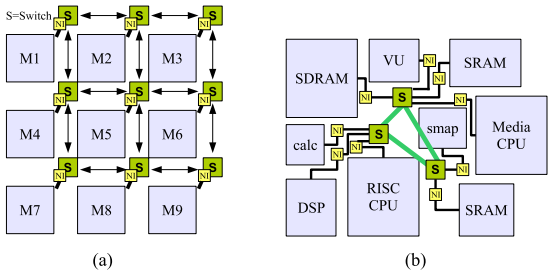


Fig. 1 Two types of NoCs. (a) Regular NoCs. (b) Application-specific NoCs.

A NoC with fewer switches will lead to longer core to switch links, causing higher link power consumption. On the other hand, when many smaller switches are used, the flows have to traverse more switches, leading to larger switch power consumption. Thus, for the NoC topology synthesis procedure, proper switch number needs to be determined, which will have a large influence on the total power consumption. Moreover, as the physical information of cores and network components (such as switches and network interfaces) also influence the link power consumption, their positions should be considered during topology generation.

A lot of works have been done to synthesize the application-specific NoC topology. In [6], a two-step topology synthesis procedure is proposed. But as the min-cut partition is carried out before floorplanning, physical infor-

Manuscript received July 17, 2011.
Manuscript revised September 0, 2011.
[†]The authors are with the Graduate School of Information Production and Systems, Waseda University, Kitakyushu, Japan.
[††]The author is with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, USA.
[†††]The author is with the Department of Computer Science & Technology, Tsinghua University, Beijing, China.
*This paper was presented at the IEEE International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 2011.
a) E-mail: E-mail: wzhong@ruri.waseda.jp
DOI: 10.1587/transele.E0.C.1

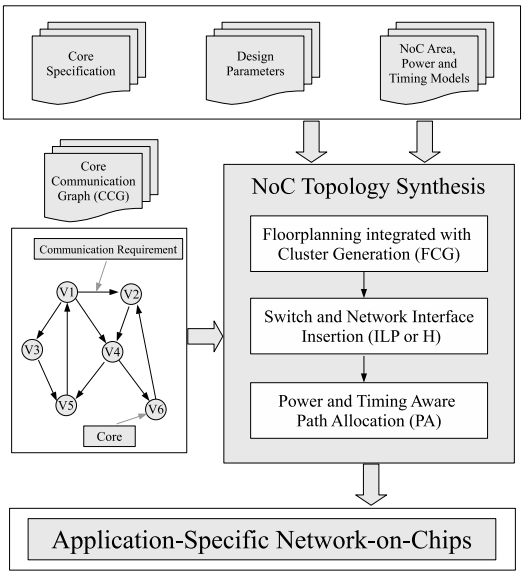


Fig. 2 NoC Design Approach Overall

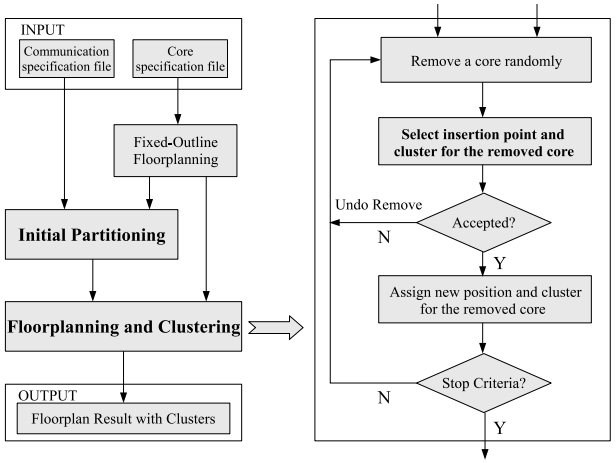


Fig. 3 Floorplanning integrated with Cluster Generation (FCG)

mation such as the distances among cores can not be taken into consideration. In [7], two heuristic algorithms are proposed to examine different set partitions. But the partition is carried out only based on communication flow and a physical network topology has to be generated for each set partition. In [8], a novel NoC topology generation algorithm is presented, however their solutions only consider topologies based on a slicing structure where switch locations are restricted to corners of cores. In [9] and [11], synthesis approaches for designing power-performance efficient NoC topology are proposed. But, physical locations of the cores are assumed as inputs and in order to obtain the optimal switch number, authors explore the designs with several different partition numbers. Moreover, as the switches located by the authors resulting in overlaps with cores, they have to reuse the floorplanner to remove the overlaps. In [12] a partition-driven floorplaning algorithm is proposed. But the authors assume optimal switch number is given as inputs, and apply min-cut partitioning every iteration in simulated annealing. Switch and network interface positions are located separately and switches are inserted into whitespace one by one. Besides, the authors use CBL [13] to represent floorplans, using lots of dummy blocks to ensure good solutions, on penalty of longer running time.

In this paper, under the consideration of both communication requirements and physical information among cores, partitioning is integrated into the floorplanning phase to explore the optimal switch number for clustering the cores with minimized link and switch power consumption. Then, an Integer Linear Programming (ILP) method is proposed for small applications to determine the optimal positions of the switches and network interfaces on the floorplan. A heuristic method (H) is also proposed for large applications by applying a two-step insertion (ILP formulation for switches and min-cost max-flow algorithm for network interfaces) to locate positions with minimized link power consumption. At last, a power and timing aware path allocation algorithm (PA) [6] is carried out to determine the connectivity across the different switches which are free of deadlock.

The rest of this paper is organized as follows. Section 2 presents the approach used for topology synthesis. Section 3 presents the FCG algorithm, which integrates partitioning into floorplanning phase. Section 4 presents an ILP formulation and a heuristic method to determine the positions of switches and network interfaces. Section 5 presents the power and timing aware path allocation algorithm. Experimental results and conclusions are presented in Sections 6 and 7, respectively.

2. Design Approach

Fig.2 shows the approach used for NoC topology synthesis. The input of the synthesis procedure is a Core Communication Graph (CCG), which could be represented by a directed graph $G = (V, E)$. Each vertex $v_i \in V$ represents a core and the edge e_{ij} with the weight w_{ij} represents the communication requirement between core c_i and c_j. In the core specification file, the name and size of different cores are obtained as inputs. In addition, NoC design parameters such as the NoC operating frequency and latency constraints are obtained. For the synthesis procedure, the area, power and timing models of the NoC switches and links are also taken as inputs.

As the topology synthesis problem is NP-Hard [14], we present efficient heuristics to synthesize the best topology for the design. Floorplanning integrated with Clustering Generation (FCG) integrates the partitioning and floorplanning to explore the optimal clustering of cores with minimized power consumption. Then, an Integer Linear Programming (ILP) and a heuristic method (H) are proposed to place switches and network interfaces on the floorplan, so that accurate power and delay can be obtained for the wires.

At last, a path allocation algorithm (PA) [6] is carried out, which takes linear combination of power consumption and hop-count as objective, to determine the connectivity across different switches.

The output of the synthesis procedure is an optimized application-specific NoC topology with pre-determined paths on network to route the traffic flows and the floorplan result of cores, switches and network interfaces in the NoC with minimized link and switch power consumption.

3. Floorplanning integrated with Cluster Generation (FCG)

Fig.3 shows the flow of the proposed algorithm. The initial solution of floorplan is generated by a fixed-outline floorplanning tool IARFP [15], which drives the floorplan with the objective evaluated by the linear combination of the area costs and the wirelength. The step ***Initial Partitionning*** will be generated based on the Core Communication Graph (CCG) by a min-cut bi-partitioning algorithm and is assumed as the input of the following ***Floorplanning and Clustering***, which integrate the partitioning and floorplanning to explore optimal clustering of cores with minimized link and switch power consumption. After floorplanning, the clusters with zero core will be ignored and the optimal switch number and connectivity between cores and switches will be determined.

In Fig.3, for the required operating frequency of the NoC, the maximum size of the switch max_sw_size is obtained as an input. ***Initial Partitionning*** apply a recursive min-cut bi-partitioning algorithm on CCG, according to the communication requirements and physical locations of the cores, until each cluster has the core number smaller than max_sw_size. In partition, we define new edge weight w'_{ij} in CCG as:

$$w'_{ij} = \alpha_w \times \frac{w_{ij}}{max\_w} + (1 - \alpha_w) \times \frac{min\_dis}{dis_{ij}} \quad (1)$$

where w_{ij} denotes communication requirement between core i and core j, dis_{ij} denotes distance between core i and j, max_w is the maximum communication requirement over all flows and min_dis is minimum distance among cores.

This step is to generate an initial partition, ensuring those cores with larger communication requirements and less distances are assigned to the same cluster and using the same switch for communication.

Once the initial partition is generated, the next step is to explore optimal clustering of cores during floorplanning. This step is carried out at ***Select insertion point and cluster for the removed core***, and the flow is listed as follows:

a. Compute the floorplan of cores except the removed one.
b. Enumerate possible insertion points based on the floorplan information obtained in step *a*, and for each insertion point, calculate the candidate cluster of the removed core by rough power evaluation.
c. Select a fixed number of candidate insertion points (CIPs) for the removed core by rough cost evaluations.
d. Choose for the removed core one of the candidate insertion points selected in step *c*, and assign the cluster for the removed core.

In step *b*, we use R_k to represent the bounding resources for cores in cluster k, and C_k represents the set of cores in kth cluster. So, if we insert the removed core c_m into (x, y), we first calculate the candidate cluster set $CCS^m(x, y)$, which is composed of clusters whose R_k covered (x, y). This step is used to ensure for every cluster $k \in CCS^m(x, y)$, the distance of the removed core c_m to all the cores in cluster k will be taken into a small range, hence can use the same switch for communication. The rough power consumption of c_m to cluster k ($k \in CCS^m(x, y)$) can be calculated as:

$$P_k^m(x, y) = \sum_{c_i \notin C_k, i \le n_c} cr\_core_{i,m} * (|x_{c_i} - x| + |y_{c_i} - y|) \quad (2)$$

where c_i denotes the ith core, $cr\_core_{i,m}$ represents communication requirements between core c_m and c_i, (x_{c_i}, y_{c_i}) is the coordinate of core c_i and n_c represents the number of cores. The rough power consumption $P^m(x, y)$ can be evaluated as:

$$P^m(x, y) = min\ \{P_k^m(x, y)\}, \forall k \in CCS^m(x, y) \quad (3)$$

the cluster k ($k \in CCS^m(x, y)$) with the smallest $P_k^m(x, y)$ will be assumed as the candidate cluster of core c_m for the insertion point (x, y), denoted as $CC^m(x, y)$, and the corresponding $P_k^m(x, y)$ will be evaluated as the $P^m(x, y)$ during the rough cost evaluation in step *c*.

In step *c*, every insertion point in Sequence-Pair with the corresponding (x, y) is evaluated by the linear combination of the area costs, wire length and rough power consumption $P^m(x, y)$ related to the removed core c_m.

In step *d*, we insert core c_m into Sequence-Pair at each CIP and evaluate all the CIPs selected in step *c* accurately:

$$\Phi = \lambda_a A + \lambda_w W + \lambda_p P + \lambda_s S \quad (4)$$

where A represent area of the floorplan; W represent the total wire lengths; P represent the total link power and S represents the switch size of candidate cluster k ($k = CC^m(x, y)$) in CIP. The total link power P can be evaluate as:

$$P = \sum_{i \le n_c} \sum_{j \ne i, j \le n_c} cr\_core_{i,j} * (|x_{c_i} - x_{c_j}| + |y_{c_i} - y_{c_j}|) \quad (5)$$

and S is involved to punish the cost if the cluster k with its size bigger than max_sw_size, which can not support the chip frequency. The parameters λ_a, λ_w, λ_p, λ_s can be used to adjust the relative weighting between the contributing factors.

If the best one of CIPs shows an improvement, the corresponding insertion point (x, y) will be the new position of the removed core c_m, and its the candidate cluster $CC^m(x, y)$ in CIP will include the core c_m. Otherwise, an acceptable probability will be calculated.

A simple example is shown in Fig.4. After the initial

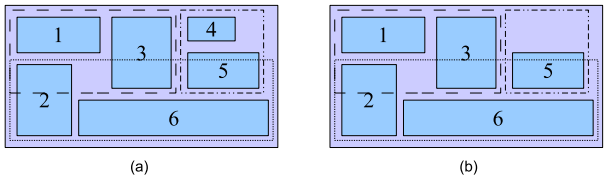


Fig. 4 Floorplan and clustering of cores. (a)Initial floorplan and partition for the cores. (b) Floorplan and bounding resources for clusters after core c_4 is removed.

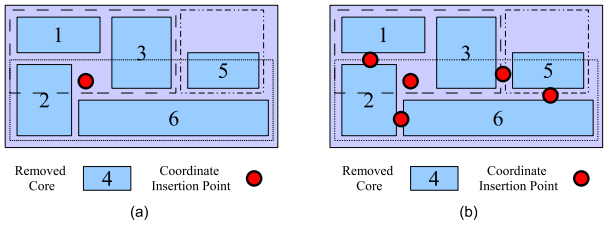


Fig. 5 Candidate insertion points calculation. (a)Candidate cluster calculation for the insertion point. (b) Fixed number of candidate insertion points.

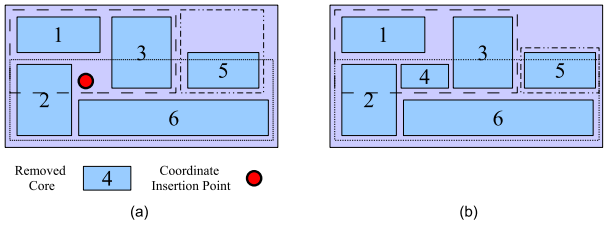


Fig. 6 Clustering of cores during floorplanning. (a)The best candidate insertion point for the removed core c_4. (b) Update the floorplan result and bounding resources for clusters after insertion of the removed core c_4.

floorplan and initial partition, six cores are placed on the floorplan and partitioned into three clusters: cluster 1 with the cores c_1 and c_3; cluster 2 with the cores c_2 and c_6; cluster 3 with the cores c_4 and c_5. We also calculate the bounding resources R_k for each cluster k, as shown in Fig.4a. Then we explore optimal clustering of cores during floorplanning. First, we remove a core (here we remove c_4) and compute the floorplan of cores except c_4. In this step, the bounding resources R_3 for cluster 3, which includes the removed core c_4, will not be changed, as shown in Fig.4b. Then, we enumerate possible insertion points for c_4, and calculate the candidate cluster of c_4 for each insertion point by rough power evaluation. In Fig.5a, the insertion point is covered by R_1 and R_2, so the candidate cluster set of c_4 for this insertion point $CCS^4(x, y)$ includes cluster 1 and cluster 2. Based on the evaluation model (2), we evaluate the rough power consumption of c_4 to cluster 1 and cluster 2, and the cluster with smaller rough power consumption will be assumed as its candidate cluster at this insertion point (here we assume as cluster 1). Then a fixed number of candidate insertion points with the corresponding clusters for the removed core c_4 are selected by rough cost evaluations, as shown in Fig.5b. Lastly, we insert the removed core c_4 into Sequence-Pair at each candidate insertion point and evaluate them accurately, using (4). If the best candidate insertion point shows an improvement, it will be the new position of the removed core c_4. As shown in Fig.6, the removed core c_4 is inserted into its best candidate insertion point on the floorplan and included by the corresponding candidate cluster, cluster 1. After insertion of the removed core c_4, the floorplan result and the bounding resources for all the clusters will be updated.

After floorplanning, the clusters with zero core will be ignored and the optimal switch number is determined. The connectivity between cores and switches is also established, which can fully support the chip operating frequency.

4. Switch and Network Interface Insertion

New switches and network interfaces will be included in the NoC topology so their physical positions must be determined to estimate the link power and delay. Due to the restriction that switches and network interfaces cannot be placed on the core, the location must be within a whitespace.

To solve this kind of problem, an even grid structure is used, whose size $P \times Q$ is determined by a specified individual grid size. Given a floorplan result, we calculate the amount of whitespace in each grid g_i, denoted as $ws(g_i)$. Let A be the area of a switch or network interface. The capacity $cap(g_i)$ of a grid g_i, i.e., the number of switches or network interfaces that can be located at g_i, is defined as $cap(g_i) = \lfloor ws(g_i)/A \rfloor$.

4.1 ILP Formulation

Instead of inserting switches and network interfaces seperately, we formulate the problem as an Integer Linear Programming (ILP) which can insert switches and network interfaces to the optimal position simultaneously with the minimized link power consumption. We want to minimize the following cost:

$$cost = P_{c2ni} + P_{ni2sw} + P_{sw2sw} \quad (6)$$

where P_{c2ni} and P_{ni2sw} denotes the power consumption between cores to network interfaces and network interfaces to switches respectively and P_{sw2sw} is the power consumption of interconnects among switches. TABLE 1 shows the notations used in the ILP formulation.

Let $a_{i,m}$ denotes whether to choose grid g_i to insert network interface ni_m and $b_{j,k}$ denotes whether to choose grid g_j to insert switch sw_k. $a_{i,m} = 1$ if grid g_i is assigned to ni_m, otherwise $a_{i,m} = 0$. $b_{j,k} = 1$ if grid g_j is assigned to sw_k, otherwise $b_{j,k} = 0$.

If grid g_i is assigned to ni_m, the sum of the Manhattan distances between a network interface ni_m and the corre-

Table 1 Notation used in ILP Formulation

n_c	number of cores(network interfaces).
n_{sw}	number of clusters(switches).
n_g	number of grids with non-zero capacity.
C_k	set of cores in kth cluster.
$CORES$	set of cores, $CORES = \{c_1 \dots c_{n_c}\}$.
c_i	the ith core ($1 \le i \le n_c$).
ni_i	the ith network interface(NI).
sw_i	the ith switch.
g_i	the ith grid.
$cap(g_i)$	capacity of the grid g_i.
(x_{c_i}, y_{c_i})	coordinate of the core c_i.
(x_{g_i}, y_{g_i})	coordinate of grid g_i.
$a_{i,m}$	whether insert ni_m into grid g_i, $a_{i,m}$=1, if insert ni_m into g_i, otherwise $a_{i,m}$=0.
$b_{j,k}$	whether insert sw_k into grid g_i, $b_{j,k}$=1, if insert sw_k into g_j, otherwise $b_{j,k}$=0.

sponding core c_m is given by:

$$dis^i_{ni_m,c_m} = |x_{g_i} - x_{c_m}| + |y_{g_i} - y_{c_m}| \tag{7}$$

where (x_{g_i}, y_{g_i}) represents the coordinate of grid g_i and (x_{c_m}, y_{c_m}) is the coordinate of the core c_m.

The distance between network interface ni_m and the corresponding core c_m is calculated as:

$$dis\_nic_m = \sum_{i=1}^{n_g} a_{i,m} \cdot dis^i_{ni_m,c_m} \tag{8}$$

Let C_k be the set of cores in the kth cluster. We have $\forall i, j, C_i \cap C_j = \phi$ and $\bigcup_{k=1}^{n_{sw}} C_k = CORES$. For each network interface ni_e with its core $c_e \in C_k$, the distance between ni_e and the switch sw_k is denoted as $dis\_nis_{e,k}$:

$$dis\_nis_{e,k} = \sum_{i=1}^{n_g}\sum_{j=1}^{n_g} a_{i,e} \cdot b_{j,k} \cdot disg_{i,j} \tag{9}$$

where $disg_{i,j}$ is the distance between grid g_i and grid g_j:

$$disg_{i,j} = |x_{g_i} - x_{g_j}| + |y_{g_i} - y_{g_j}| \tag{10}$$

The distance between switch sw_d and switch sw_t is denoted as $dis\_sw_{d,t}$:

$$dis\_sw_{d,t} = \sum_{i=1}^{n_g}\sum_{j=1}^{n_g} b_{i,d} \cdot b_{j,t} \cdot disg_{i,j} \tag{11}$$

However, the equation for $dis\_nis_{e,k}$ and $dis\_sw_{d,t}$ above are illegal in an ILP because they are non-linear. As a result, we introduce boolean variables $\lambda_{ie,jk}$ and $\gamma_{id,jt}$ to replace $a_{i,e} \cdot b_{j,k}$ and $b_{i,d} \cdot b_{j,t}$, respectively, and enforce the following artificial constraints in our ILP:

$$\begin{aligned} dis\_nis_{e,k} &= \sum_{i=1}^{n_g}\sum_{j=1}^{n_g} \lambda_{ie,jk} \cdot disg_{i,j} \\ a_{i,e} + b_{j,k} - \lambda_{ie,jk} &\le 1 \\ a_{i,e} - \lambda_{ie,jk} &\ge 0 \\ b_{j,k} - \lambda_{ie,jk} &\ge 0 \end{aligned} \tag{12}$$

Because of constraints(12), and the fact that $dis\_nis_{e,k}$ appears in the cost function to be minimized, $\lambda_{ie,jk}$ will be equal to 0 unless both $a_{i,e}$ and $b_{j,k}$ are 1. Similarly, $dis\_sw_{d,t}$ can be re-written as:

$$\begin{aligned} dis\_sw_{d,t} &= \sum_{i=1}^{n_g}\sum_{j=1}^{n_g} \gamma_{id,jt} \cdot disg_{i,j} \\ b_{i,d} + b_{j,t} - \gamma_{id,jt} &\le 1 \\ b_{i,d} - \gamma_{id,jt} &\ge 0 \\ b_{j,t} - \gamma_{id,jt} &\ge 0 \end{aligned} \tag{13}$$

Let cr_core_m be the communication requirement of the core c_m, and $cr\_sw2sw_{d,t}$ be the communication requirement between switch sw_d and switch sw_t. To minimize the total power consumption of the links, we need to minimize the length of the links weighted by their communication requirement values, so that higher communication requirements are shorter than lower ones. Formulating the objective function mathematically, we get:

$$\begin{aligned} cost = &\sum_{m=1}^{n_c} dis\_nic_m \cdot cr\_core_m \\ &+ \sum_{k=1}^{n_{sw}} \sum_{c_e \in C_k} dis\_nis_{e,k} \cdot cr\_core_e \\ &+ \sum_{d=1}^{n_{sw}} \sum_{t \ne d}^{n_{sw}} dis\_sw_{d,t} \cdot cr\_sw2sw_{d,t} \end{aligned} \tag{14}$$

The ILP formulation for optimizing switch and network interface positions is as follows:

$$\begin{aligned} &minimize \;\; cost \\ &subject\ to \;\; Equations(7) - (14) \\ &\sum_{i=1}^{n_g} a_{i,e} = 1, \;\; \forall e \in \{1 \dots n_c\} \\ &\sum_{j=1}^{n_g} b_{j,k} = 1, \;\; \forall k \in \{1 \dots n_{sw}\} \\ &\sum_{e=1}^{n_c} a_{i,e} + \sum_{k=1}^{n_{sw}} b_{i,k} \le cap(g_i), \;\; \forall i \in \{1 \dots n_g\} \\ &a_{i,e}, b_{j,k}, \lambda_{ie,jk}, \gamma_{id,jt} = 0 \; or \; 1 \end{aligned} \tag{15}$$

We adopted *Cbc* [16] as our ILP solver to obtain the optimum solutions. For small applications (12 cores, 3 switches), the optimal solution can be obtained in few seconds.

4.2 Hierarchical ILP Flow for Accelerating

Computationally, ILP is one of the known NP-hard problems [17], it will be very time-consuming for large applications. Thus, in this subsection, we use a hierarchical

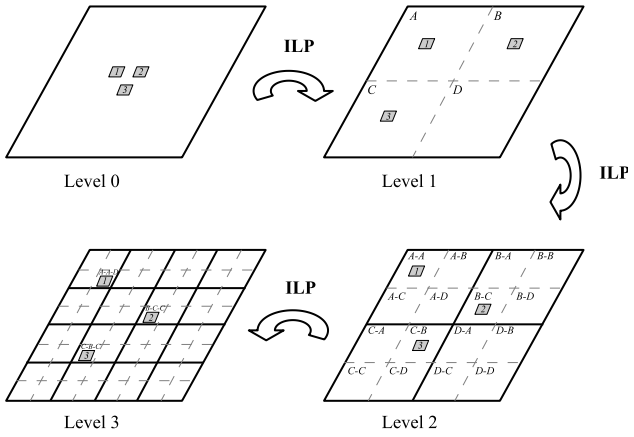


Fig. 7 Example of Hierarchical ILP Flow for Accelerating

flow to deal with the problem, locating network components (switches and network interfaces) efficiently. We use the following hierarchical flow to reduce the number of variables in ILP formulation:

a. Divide the chip region equally into some subregions.
b. Globally locate the network components into the subregions with non-zero capacity by applying Integer Linear Programming (ILP).
c. Divide each subregion equally into some smaller subregions. Each smaller subregion will be assumed as the candidate positions for the network components which are located into the corresponding upper-level subregion. Then, apply *b*.
d. Recursively apply *c* until each subregion contains small number of grids with non-zero capacity.
e. On the bottom level of hierarchical flow, based on the reduced number of possible grids for each network component, apply Integer Linear Programming (ILP) to insert switches and network interfaces into the grids simultaneously.

Fig.7 shows an example of a 3-level hierarchical ILP flow. On level 0, initially, all the network components (switches and network interfaces) are located at the center of the chip (here we use switch 1, 2, 3 as example). Then the chip is divided into four subregions: A, B, C and D. These four subregions are assumed as candidate positions to locate the three switches. On level 1, we apply Integer Linear Programming (ILP) globally to insert the switches into the four subregions. In this example, switches 1, 2 and 3 are inserted into subregions A, B and C respectively. Then, on level 2, we divide each subregion into four small subregions. For example, as is shown, subregion A is divided into four small subregions A-A, A-B, A-C and A-D. Thus, these four small subregions are assumed as candidate positions to locate switch 1. We apply ILP globally, inserting switches 1, 2 and 3 into subregions A-A, B-C and C-B respectively. On level 3, we also divide each subregion into four small subregions and apply ILP globally. As is shown in Fig.7, finally, switches 1, 2 and 3 are inserted into subregions A-A-D, B-C-C and C-B-C respectively.

4.3 Heuristic Algorithm

Instead of inserting switches and network interfaces simultaneously, we propose two exact methods to insert switches and network interfaces separately, which will be very fast for large applications. We notice that, even for large applications, the switch number will be small. Hence, the switch insertion problem is formulated as an Integer Linear Programming solved by the ILP solver *Cbc*. And network interface insertion problem is formulated as a min-cost max-flow problem.

4.3.1 Switch Insertion

In [12], authors insert switches one by one. Here, we formulate switch insertion problem as an Integer Linear Program (ILP) which can insert switches simultaneously to minimize link power consumption between switches. The objective is to minimize the following cost:

$$cost = P_{c2sw} + P_{sw2sw} \tag{16}$$

where P_{c2sw} denotes power consumption of interconnects between cores to the corresponding switches.

If switch sw_k is assigned into grid g_j, the distances from core $m \in C_k$ to switch sw_k is denoted as $dis\_cs^j_{m,k}$.

$$dis\_cs^j_{m,k} = |x_{g_j} - x_{c_m}| + |y_{g_j} - y_{c_m}| \tag{17}$$

So, we formulate the objective function mathematically and we get:

$$\begin{aligned} cost = &\sum_{k=1}^{n_{sw}} \sum_{j=1}^{n_g} b_{j,k} \cdot \sum_{m \in C_k} dis\_cs^j_{m,k} \cdot cr\_core_m \\ &+ \sum_d \sum_t dis\_sw_{d,t} \cdot cr\_sw2sw_{d,t} \end{aligned} \tag{18}$$

The ILP formulation for optimize switch positions is written as follow:

$$\begin{aligned} &minimize \;\; cost \\ &subject\ to \;\; Equations(10), (11), (13), (17) - (18) \\ &\qquad \sum_{i=1}^{n_g} b_{i,k} = 1, \;\; \forall k \in \{1 \ldots n_{sw}\} \\ &\qquad \sum_{k=1}^{n_{sw}} b_{i,k} <= cap(g_i), \;\; \forall i \in \{1 \ldots n_g\} \\ &\qquad b_{i,k}, \gamma_{id,jt} = 0 \; or \; 1 \end{aligned} \tag{19}$$

4.3.2 Network Interface Insertion

Once the switch positions are obtained, the next step is to

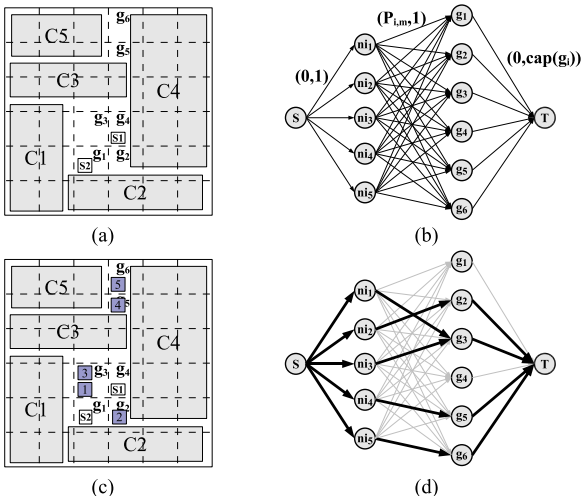


Fig. 8 A sample of NIs insertion. (a)The candidate grids (*GRIDS*) for NIs insertion are the grids $g_1, g_2, g_3, g_4, g_5, g_6$. (b) Corresponding network flow model. (c) After min-cost max-flow algorithm, NIs insert into grids 3,2,3,5,6 respectively. (d) Corresponding network flow result.

find the optimal positions of network interfaces. Previous work [12] carried out min-cost max-flow algorithm to assign network interfaces into grids. We also use this method to locate network interfaces but introduce a more accurate link power evaluation model other than the distance between each network interface to the corresponding switch.

When insert a network interface ni_m of $c_m(\in C_k)$ into grid g_i, the distance between ni_m to the core c_m and switch sw_k can be calculated as:

$$dis^i_{ni_m} = dis^i_{ni_m,c_m} + (|x_{g_i} - x_{sw_k}| + |y_{g_i} - y_{sw_k}|) \tag{20}$$

where $dis^i_{ni_m,c_m}$ is the distance between ni_m and c_m defined in Equation(7), and (x_{sw_k}, y_{sw_k}) is the coordinate of the switch sw_k. We define communication requirement of core c_m as cr_core_m, and the power consumption for inserting ni_m to grid g_i is evaluated as:

$$P_{i,m} = cr\_core_m \cdot dis^i_{ni_m} \tag{21}$$

Let NI represents the set of network interfaces and $GRIDS$ represents the set of grids with non-zero capacity. For each $g_i \in GRIDS$, its capacity is denoted as $cap(g_i)$. We construct a network graph $G = (V, E)$, and use a min-cost max-flow algorithm to determine the positions of network interfaces with minimized total link power consumption. A simple example is shown in Fig.8.

- $V = \{s, t\} \cup NI \cup GRIDS$.
- $E = \{(s, ni_m)|ni_m \in NI\} \cup \{(ni_m, g_i)|\forall g_i \in GRIDS\} \cup \{(g_i, t)|g_i \in GRIDS\}$.
- Capacities: $C(s, ni_m) = 1, C(ni_m, g_i) = 1, C(g_i, t) = cap(g_i)$.
- Cost: $F(s, ni_m) = 0, F(ni_m, g_i) = P_{i,m}, F(g_i, t) = 0$.

Network interface insertion can be done efficiently by min-cost max-flow algorithms running in polynomial time [18].

Table 2 Power Consumption of Switches

Ports (in x out)	2x2	3x2	3x3	4x3	4x4	5x4	5x5
Leakage power (W)	0.0069	0.0099	0.0133	0.0172	0.0216	0.0260	0.0319
Bit energy (pJ/bit)	0.3225	0.0676	0.5663	0.1080	0.8651	0.9180	1.2189

Table 3 Power Consumption of Links

Wire length (mm)	1	4	8	12	16
Leakage power (W)	0.000496	0.001984	0.003968	0.005952	0.007936
Bit energy (pJ/bit)	0.6	2.4	4.8	7.2	9.6

5. Power and Timing Aware Path Allocation

During the procedure of establishing physical links and paths for traffic flows, we take linear combination of power consumption and hop-count as objective. In this procedure, the flows are ordered in decreasing rate requirements, and the bigger flow are assigned first by applying Dijastra's shortest path algorithm. When opening a new physical link, we also check whether the switch size is small enough to satisfy the particular frequency of operation. In [6] and [19], the authors present methods to remove both routing and message dependent deadlocks when computing the paths. We also use the methods to obtain paths that are free of deadlock.

6. Experimental Result

The proposed methods have been implemented in C++ language and run on an IBM workstation (3.2 GHz and 3GB RAM) with Linux OS. We use hMetis [20] as our partitioning tool to generate the initial partition. Besides, we adopted *Cbc* [16] as our ILP solver.

6.1 Method of Power Evaluation

In NoC architecture, the total power consumption includes dynamic power and the related leakage power. The power consumption can be calculated as[†]:

$$P = \sum_{i \in NL} (E^i_l * cr_i + lP^i_l) + \sum_{k \in SW} (E^k_s * cr_k + lP^k_s) \tag{22}$$

where NL and SW represents the set of network links and switches respectively. E^i_l and E^k_s are the bit energy of link i and switch k respectively. cr_i and cr_k denotes the communication requirements passing on link i and switch k. The leakage power of link i and switch k are denoted as lp^i_l and lp^k_s respectively. The leakage power and bit energy of switches with different example port configurations in 70nm technology are showed in TABLE 2. Power consumption of links is listed in TABLE 3. The power consumption is estimated using power simulator Orion [21].

[†]Here we ignore the internal power consumption of cores and

Table 4 NoC Synthesis Results for small applications

Benchmark	V#	E#	Part#			Power(mW)					Hop Count			Time(s)				
			PDF	FIP	FHP	PDF	FIP	IMP(%)	FHP	IMP(%)	PDF	FIP	FHP	PDF	FIP	IMP(%)	FHP	IMP(%)
MPEG4	12	13	3	3	3	52.2	21.17	-59.44	24.21	-53.62	1.16	1	1	10.54	3.32	-68.5	0.42	-96.02
MWD	12	12	3	2.8	2.8	7.93	6.89	-13.11	7.23	-8.8	1.33	1.16	1	10.61	6.47	-39.02	0.44	-95.85
VOPD	12	14	3	2.4	2.4	35.61	24.31	-31.73	27.62	-22.44	1	1	1	11.02	5.48	-50.27	0.37	-96.64
263decmp3dec	14	15	3	3.6	3.6	153.86	126.79	-17.59	138.03	-10.29	1	1	1.14	17.12	4.73	-72.37	0.51	-97.02
263encmp3dec	12	12	3	3	3	1885.1	1590.1	-15.65	1618.23	-14.16	1	1.07	1.06	9.92	2.1	-78.83	0.44	-95.56
mp3encmp3dec	13	13	3	3.2	3.2	164.89	119.19	-27.72	131.93	-20	1	1	1	15.17	1.98	-86.95	0.42	-97.23
Avg	-	-	-	-	-	-	-	-27.54%	-	-21.55%	1.08	1.04	1.03	-	-	-66%	-	-96.39%

Table 5 NoC Synthesis Results for large applications

Benchmark	V#	E#	Part#			Power(mW)					Hop Count			Time(s)				
			PDF	FIP-H	FHP	PDF	FIP-H	IMP(%)	FHP	IMP(%)	PDF	FIP-H	FHP	PDF	FIP-H	IMP(%)	FHP	IMP(%)
D_38_tvopd	38	47	3	8	8	147.96	101.08	-31.68	91.27	-38.3	1.33	1.01	1.03	112.81	26.18	-76.79	11.58	-89.73
D_36	36	43	3	8	8	289.69	216.04	-25.42	215.09	-25.75	1.33	1.03	1.03	191.37	19.03	-90.06	10.93	-94.29
D_43	43	54	3	9	9	454.1	297.45	-34.5	296.29	-34.75	1.33	1.03	1.05	608.95	23.24	-96.18	10.91	-98.21
D_50	50	57	3	12	12	225.8	184.87	-18.13	161.98	-28.26	1.33	1.06	1.06	784.09	35.16	-95.52	43.15	-94.5
Avg	-	-	-	-	-	-	-	-27.43%	-	-31.77%	1.33	1.03	1.04	-	-	-89.64%	-	-94.18%

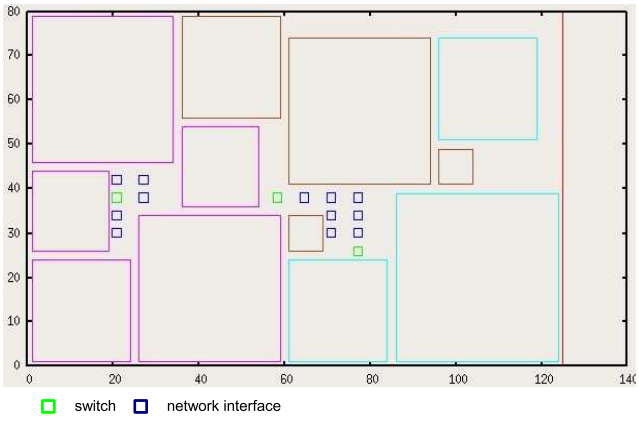


Fig. 9 Floorplan of MPEG4 with switches and network interfaces.

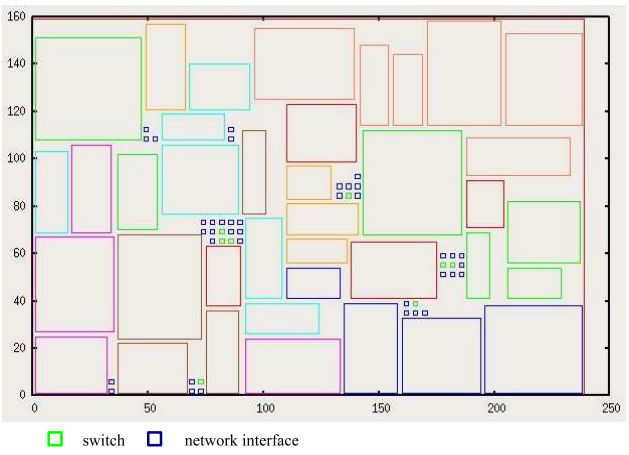


Fig. 10 Floorplan of D_38_tvopd with switches and network interfaces.

6.2 Results and Discussion

Four sets of benchmarks are used to evaluate the proposed algorithm. The first set of benchmarks are three video processing applications obtained from [22], including VOPD, MPEG4, and MWD. The next set of benchmarks are obtained from [23], including 263decmp3dec, 263encmp3dec and mp3encmp3dec. The benchmark D_38_tvopd is obtained from [9]. Finally, we generate several larger synthetic benchmarks from the above applications.

Fig.9 shows the floorplan results of the cores and network components for MPEG4, generated by our FIP (FCG+ILP+PA) method and Fig.10 shows the floorplan results for D_38_tvopd based on our FHP (FCG+H+PA) method.

We compared the proposed method with another three-stage synthesis approach PDF [12], which applies a partition-driven floorplanning based on a given switch number and, in the second stage, places switches and network interfaces separately on the floorplan. The authors also carry out a power and timing aware algorithm as its third stage for path allocation. The data are averages of 10 runs.

TABLE 4 shows the comparison of the topologies synthesized by the proposed method and PDF. The column Power means the actual power consumption and Hop Count means average number of hops. FIP means the FCG algorithm combined with an Integer Linear Programming (ILP) to insert switches and network interfaces simultaneously,

network interfaces as they are constant and will not change with their positions in the NoC topology.

and a power and timing aware path allocation algorithm (PA). FHP combined FCG with the heuristic method (H) and path allocation algorithm (PA). IMP shows the improvement of the proposed method. In PDF, partition number (=3) is given as an input and switches and network interfaces are inserted separately. Compared with PDF, FIP (FCG+ILP+PA) synthesis method can save 27.54% of power, 4% of hop-count and 66% of running time on average. The heuristic method FHP (FCG+H+PA) also saves 21.55% power, 5% of hop-count and 96.39% of running time on average. As FIP integrates the partitioning and floorplanning to explore the optimal clustering of cores, and inserts switches and network interfaces simultaneously, a significant power and hop-count reduction could be achieved. Moreover, PDF applies min-cut partitioning every iteration in simulated annealing, and uses CBL [13] as the floorplan representation, which uses lots of dummy blocks to ensure good solutions on penalty of longer running time. On the other hand, FIP applies a recursive min-cut bi-partitioning algorithm only once to generated an initial partition and adjusts the clustering of cores during floorplanning (based on a very fast floorplanner IARFP [15], Sequence-Pair representation), a large reduction of running time could be achieved.

For further demonstrating the effectiveness, we carried out FIP-H, (FCG+ILP+PA) with hierarchical ILP flow for accelerating and FHP (FCG+H+PA) method for large applications. As is shown in TABLE 5, for large applications, compared with PDF, FIP-H synthesis method can save 27.43% of power, 30% of hop-count and 89.64% of running time on average. Also, for large applications, such as D_43, compared with PDF, FHP reduces power consumption from 454.1 mW to 296.29 mW, hop-count from 1.33 to 1.05 and running time from 608.95 s to 10.91 s. Generally, 31.77% of power consumption, 29% of hop-count and 94.18% of running time can be saved for large applications base on FHP method.

7. Acknowledgements

This research was supported by a grant of Knowledge Cluster Initiative 2nd stage implemented by Ministry of Education, Culture, Sports, Science and Technology(MEXT) and CREST (Core Research for Evolutional Science and Technology) JST, Japan.

8. Conclusions

In this paper, a FCG algorithm is proposed, which integrate the partitioning and floorplanning to explore optimal clustering of cores with minimized power consumption. For small applications, an Integer Linear Programming (ILP) method is proposed to place switches and network interfaces optimally on the floorplan, so that accurate power and delay are obtained for the wires. For large applications, the proposed heuristic algorithm is also efficient for switches and network interfaces insertion. Experimental results show that our NoC topology leads to a large reduction in power consumption, hop-count and running time. In future, we plan to extend the synthesis approach to three-dimension which needs to meet the TSV constraints and technology requirements in 3-D NoCs.

References

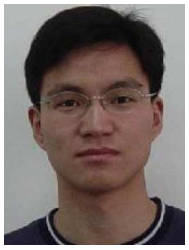
[1] W.J.Dally and B.Towles, "Route packet, not wires: on-chip Interconnection Networks," Proc.IEEE/ACM Design Automation Conference, pp.684-689, 2001.
[2] L.Benini and G.De Micheli, "Networks on chips: A new SoC paradigm," Computer, pp.70-78, 2002.
[3] J.Hu and R.Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," Proc.IEEE/ACM Asia and South Pacific Design Automation Conference, pp.233-239, 2003.
[4] S.Murali and G.De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," Proc.IEEE Design Automation and Test in Europe Conference, vol.2, pp.896, 2004.
[5] M.B.Taylor, J.Kim, J.Miller, D.Wentzlaff, F.Ghodrat, B.Greenwald, H.Hoffman, P.Johnson, Lee Jae-Wook, W.Lee, A.Ma, A.Saraf, M.Seneski, N.Shnidman, V.Strumpen, M.Frank, S.Amarasinghe, and A.Agarwal, "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," Proc.IEEE Micro, pp.25-35, 2002.
[6] S.Murali, P.Meloni, F.Angiolini, D.Atienza, S.Carta, L.Benini, G.De Micheli, and L.Raffo, "Designing Application-Specific Networks on Chips with Floorplan Information," Proc.IEEE/ACM International Conference on Computer-Aided Design, pp.355-362, 2006.
[7] S.Yan and B.Lin, "Application-specific Network-on-Chip architecture synthesis based on set partitions and Steiner Trees," Proc.IEEE/ACM Asia and South Pacific Design Automation Conference, pp.277-282, 2008.
[8] K.Srinivasan, K.S.Chatha and G.Konjevod, "Linear programming based techniques for synthesis of network-on-chip architectures," IEEE Trans. VLSI, vol.14, pp.407-420, 2006.
[9] S.Murali, C.Seiculescu, L.Benini, and G.De Micheli, "Synthesis of Networks on Chips for 3D Systems on Chips," Proc.IEEE/ACM Asia and South Pacific Design Automation Conference, pp.242-247, 2009.
[10] D.Bertozzi, A.Jalabert, S.Murali, R.Tamhankar, S.Stergiou, L.Benini, G.De Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," IEEE Trans. Parallel and Distributed Systems, vol.16, pp.113, 2005.
[11] C.Seiculescu, S.Murali, L.Benini, G.De Micheli, "SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chips," Proc.IEEE Design Automation and Test in Europe Conference, pp.9-14, 2009.
[12] B.Yu, S.Dong, S.Chen, S.Goto, "Floorplanning and Topology Generation for Application-Specific Network-on-Chip," Proc.IEEE/ACM Asia and South Pacific Design Automation Conference, pp.535-540, 2010.
[13] X.Hong, G.Huang, Y.Cai, J.Gu, S.Dong, C.Cheng, J.Gu, "Corner block list: an effective and efficient topological representation of non-slicing floorplan," Proc.IEEE/ACM International Conference on Computer-Aided Design, pp.8-12, 2000.
[14] A.Pinto, L.P.Carloni, A.L.Sangiovanni-Vincentelli, "Efficient Synthesis of Networks on Chip", Proc.IEEE International Conference on Computer Design, pp.146-150, 2003.
[15] S.Chen, T.Yoshimura, "Fixed-Outline Floorplanning: Block-Position Enumeration and a New Method for Calculating Area Costs," IEEE Trans. Computer Aided Design, pp.858-871, 2008.
[16] Cbc, "ILP solver" http://projects.coin-or.org/Cbc
[17] M.R.Garey and D.S.Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," WH Freeman and Company, San Francisco, 1979.

[18] R.K.Ahuja, T.L.Magnanti, and J.B.Orlin, ”Network Flows: Theory, Algorithms, and Applications,” Prentice Hall/Pearson, 2005.
[19] K.Goossens, A.Radulescu, A.Hansson, ”A Unified Approach to Constrained Mapping and Routing on Network-on-Chip Architectures,” Proc.IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis, pp.75-80, 2005.
[20] G.Karypis, R.Aggarwal, V.Kumar, and S.Shekhar, ”Multilevel Hypergraph Partitioning: Application in VLSI Domain,” Proc.IEEE/ACM Design Automation Conference, pp.526-529, 1997.
[21] H.Wang, X.Zhu, L.Peh, and S.Malik, ”Orion: A Power-Performance Simulator for Interconnection Networks,” Proc.IEEE/ACM International Symposium on Microarchitecture, pp.294-305, 2002.
[22] 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 Trans. Parallel and Distributed Systems, pp.113-129, 2005.
[23] K.Srinivasan, K.S.Chatha, and G.Konjevod, ”Linear programming based techniques for synthesis of network-on-chip architectures,” IEEE Trans. VLSI, pp.407-420, 2006.

Wei Zhong received the B.S. degree in Embedded System Engineering from Dalian University of Technology, China in 2008, the M.S. degree in System LSI Engineering from Waseda University, Japan in 2010. He is currently pursuing the Ph.D. degree in System LSI Design Automation Lab, Waseda University, Japan. His research interests include TSV assignment for 3D ICs, floorplanning algorithms and synthesis of networks on chips.

Takeshi Yoshimura received B.E., M.E., and Dr. Eng. degrees from Osaka University, Osaka, Japan, in 1972, 1974, and 1997. He joined the NEC Corporation, Kawasaki, Japan, in 1974, where he has been engaged in research and development efforts devoted to computer application systems for communication network design, hydraulic network design, and VLSI CAD. From 1979 to 1980 he was on leave at the Electronics Research Laboratory, University of California, Berkeley, where he worked on very large scale integration computer-aided design layout. He received Best Paper Awards from the Institute of Electronics, Information and Communication Engineers of Japan (IEICE) and the IEEE CAS Society. Dr. Yoshimura is a Member of the IEICE, IPSJ (the Information Processing Society of Japan), and IEEE.

Bei Yu received the B.S. degree in the Department of Mathematic from UESTC, China in 2007, the M.S. degree in EDA lab, Department of Computer Science and Technology, Tsinghua University, China in 2010. He is currently a Ph.D candidate in VLSI Design Automation lab, Department of Electrical and Computer Engineering, The University of Texas at Austin, USA. His research interests include CAD for VLSI, floorplanning algorithms and low power design.



Song Chen received the B.S. degree in computer science from Xi’an Jiaotong University, Xi’an, China, in 2000, the M.S. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 2003 and 2005, respectively. He is currently an assistant professor at the Graduate Schoolof IPS, Waseda University, Japan. His research interests include several aspects of electronic design automation, e.g., floorplanning, placement and high-level synthesis.

Sheqin Dong received the B.E. degree in Computer Science in 1985, M.S. degree in semiconductor physics and device in 1988, and Ph.D. degree in mechantronic control and automation in 1996. He is currently an associate professor of the EDA lab at the department of computer science and technology in Tsinghua University. His current research interests include CAD for VLSI, floorplanning and placement algorithms, multimedia ASIC and hardware design.

Satoshi GOTO received the B.E. and M.E. degree in Electronics and Communication Engineering from Waseda University in 1968 and 1970, respectively. He also received the Dr. of Engineering from Waseda University in 1981. He is IEEE fellow, Member of Academy Engineering Society of Japan and professor of Waseda University. His research interests include LSI System and Multimedia System.