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

Wei Zhong†, Bei Yu‡, Song Chen†, Takeshi Yoshimura†, Sheqin Dong‡, and Satoshi Goto†
†Graduate School of Information Production and Systems, Waseda University, Kitakyushu, Japan,
‡Department of Computer Science & Technology, Tsinghua University, Beijing, China,
†{wzhong@ruri., chensong@aoni., t-yoshimura@, goto@}waseda.jp
‡{b-yu07@mails., dongsq@mail.}tsinghua.edu.cn

Abstract[1]

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, 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.

Keywords

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, instead of the traditional non-scalable buses, on-chip micro-networks are used to interconnect the various cores. NoCs have better modularity and design predictability when compared to bus based systems.

NoCs can be utilized as regular architectures like a mesh or torus, or application-specific architectures. Regular NoC architectures offer lower design time, and are useful when implemented in a generic multiprocessor environment such as the MIT RAW [3]. On the other hand, application-specific NoC architectures consist of heterogeneous cores and memory elements which have vastly different sizes. The application-specific NoC architecture has been demonstrated to be superior to regular architectures in terms of power, area and performance [4]. This paper concentrates on the synthesis method of application-specific NoC topologies.

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 [5], a two-step topology synthesis procedure is proposed. But as the min-cut partition is carried out before floorplanning, physical information such as the distances among cores can not be taken into consideration. In [6], 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 [7], a novel NoC topology generation algorithms were presented, however their solutions only consider topologies based on a slicing structure where switch locations are restricted to corners of cores. In [8] and [9], synthesis approaches for designing power-performance efficient NoC topology are proposed. But, the physical locations of the cores are assumed as inputs and in order to obtain the optimal switch number, the 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 [10] a partition-driven floorplaning algorithm is proposed. But the authors assume optimal switch number is given as an input, and apply min-cut partitioning every iteration in simulated annealing. The switch and network interface positions are located separately and switches are inserted into whitespace one by one. Besides, the authors use CBL [11] to represent floorplans, which uses 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) [5] 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

[1]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.

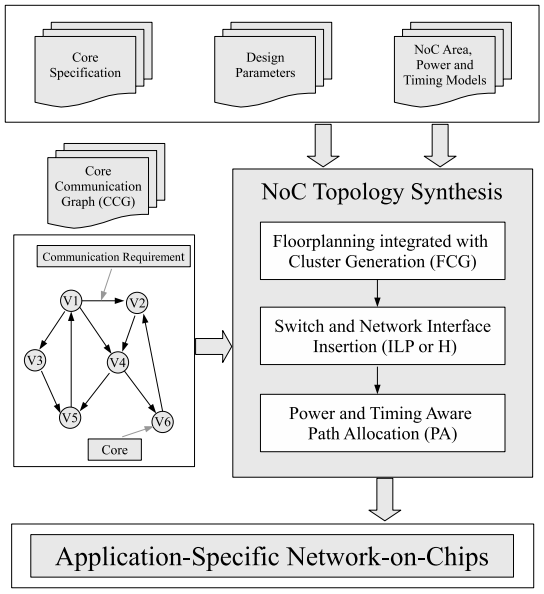

Fig. 1. NoC Design Approach Overall

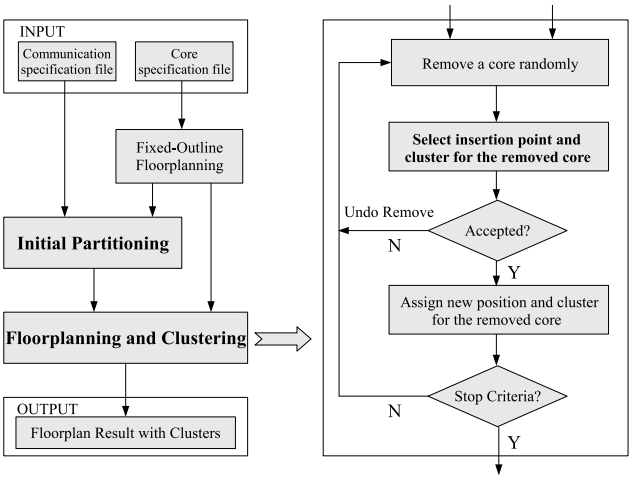

Fig. 2. Floorplanning integrated with Cluster Generation (FCG)

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.1 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 [12], 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) [5] 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 power consumption.

3. Floorplanning integrated with Cluster Generation

Fig.2 shows the flow of the proposed algorithm. The initial solution of floorplan is generated by a fixed-outline floorplanning tool IARFP [13], 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.2, 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:

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

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:

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

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:

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

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.

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.

TABLE I
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 ($1 \leq i \leq n_c$).
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.

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 separately, 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 I 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 corresponding core c_m is given by:

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

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:

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

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

$$\begin{aligned} dis\_sw_{d,t} &= \sum_{i=1}^{n_g}\sum_{j=1}^{n_g} \gamma_{id,jt} \cdot disg_{i,j} \\ b_{i,d} + b_{j,t} - \gamma_{id,jt} &\leq 1 \\ b_{i,d} - \gamma_{id,jt} &\geq 0 \\ b_{j,t} - \gamma_{id,jt} &\geq 0 \end{aligned} \tag{12}$$

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

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:

$$\begin{aligned} cost \quad = \quad & \sum_{m=1}^{n_c} dis\_nic_m \cdot cr\_core_m \\ & + \sum_{k=1}^{n_{sw}} \sum_{c_e \in C_k} dis\_nis_{e,k} \cdot cr\_core_e \\ & + \sum_{d=1}^{n_{sw}} \sum_{t \neq d}^{n_{sw}} dis\_sw_{d,t} \cdot cr\_sw2sw_{d,t} \end{aligned} \tag{13}$$

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

$$\begin{aligned} & minimize \;\; cost \\ & subject\ to \;\; Equations(7) - (13) \\ & \sum_{i=1}^{n_g} a_{i,e} = 1, \quad \forall e \in \{1 \dots n_c\} \\ & \sum_{j=1}^{n_g} b_{j,k} = 1, \quad \forall k \in \{1 \dots n_{sw}\} \\ & \sum_{e=1}^{n_c} a_{i,e} + \sum_{k=1}^{n_{sw}} b_{i,k} \leq cap(g_i), \quad \forall i \in \{1 \dots n_g\} \\ & a_{i,e}, b_{j,k}, \lambda_{ie,jk}, \gamma_{id,jt} = 0 \ or \ 1 \end{aligned} \tag{14}$$

We adopted Cbc [14] 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. However, computationally, ILP is one of the known NP-hard problems, it will be very time-consuming for large applications.

4.2 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.

4.2.1 Switch Insertion

In [10], 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:

$$cost = P_{c2sw} + P_{sw2sw} \tag{15}$$

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 $dis\_cs^j_{m,k}$.

$$dis\_cs^j_{m,k} = |x_{g_j} - x_{c_m}| + |y_{g_j} - y_{c_m}| \tag{16}$$

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

$$\begin{aligned} cost \; = \; & \sum_{k=1}^{n_{sw}}\sum_{j=1}^{n_g} b_{j,k} \cdot \sum_{m \in C_k} dis\_cs^j_{m,k} \cdot cr\_core_m \\ & + \sum_{d}\sum_{t} dis\_sw_{d,t} \cdot cr\_sw2sw_{d,t} \end{aligned} \tag{17}$$

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

$$\begin{aligned} & minimize \;\; cost \\ & subject\ to \;\; Equations(10), (11), (12), (16) - (17) \\ & \sum_{i=1}^{n_g} b_{i,k} = 1, \quad \forall k \in \{1 \dots n_{sw}\} \\ & \sum_{k=1}^{n_{sw}} b_{i,k} <= cap(g_i), \quad \forall i \in \{1 \dots n_g\} \\ & b_{i,k}, \gamma_{id,jt} = 0 \ or \ 1 \end{aligned} \tag{18}$$

4.2.2 Network Interface Insertion

Once the switch positions are obtained, the next step is to find the optimal positions of network interfaces. Previous work [10] 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:

$$dis^i_{ni_m} = dis^i_{ni_m,c_m} + (|x_{g_i} - x_{sw_k}| + |y_{g_i} - y_{sw_k}|) \quad (19)$$

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:

$$P_{i,m} = cr\_core_m \cdot dis^i_{ni_m} \quad (20)$$

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 $cap(g_i)$. The network graph $G = (V, E)$ is constructed as follows:

- $V = \{s, t\} \cup NI \cup GRIDS$.
- $E = \{(s, ni_m)|ni_m \in NI\} \cup \{(ni_m, g_i)|\forall g_i \in GRIDS\} \cup \{(g_i, t)|g_i \in GRIDS\}$.
- 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 [15].

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 [5] and [16], 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 [17] as our partitioning tool to generate the initial partition. Besides, we adopted Cbc [14] 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[2]:

$$P = \sum_{i \in NL} (E^i_l * f * cr_i + lP^i_l) + \sum_{k \in SW} (E^k_s * f * cr_k + lP^k_s) \quad (21)$$

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. f is the operating frequency. 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 II. Power consumption of links is listed in TABLE III. The power consumption is estimated using power simulator Orion [18].

[2]Here we ignore the internal power consumption of cores and network interfaces as they are constant and will not change with their positions in the NoC topology.

TABLE II
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 III
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

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 [19], including VOPD, MPEG4, and MWD. The next set of benchmarks are obtained from [20], including 263decmp3dec, 263encmp3dec and mp3encmp3dec. The benchmark D_38_tvopd is obtained from [8]. Finally, we generate several larger synthetic benchmarks from the above applications.

We compared the proposed method with another three-stage synthesis approach PDF [10], 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 IV 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, 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 [11] 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 [13], Sequence-Pair representation), a large reduction of running time could be achieved.

For further demonstrating the effectiveness, we carried out FHP (FCG+H+PA) method for large applications. As is shown in TABLE V, 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.

TABLE IV
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 V
NOC SYNTHESIS RESULTS FOR LARGE APPLICATIONS

Benchmark	V#	E#	Part#		Power(mW)			Hop Count		Time(s)		
			PDF	FHP	PDF	FHP	IMP(%)	PDF	FHP	PDF	FHP	IMP(%)
D_38_tvopd	38	47	3	8	147.96	91.27	-38.3	1.33	1.03	112.81	11.58	-89.73
D_36	36	43	3	8	289.69	215.09	-25.75	1.33	1.03	191.37	10.93	-94.29
D_43	43	54	3	9	454.1	296.29	-34.75	1.33	1.05	608.95	10.91	-98.21
D_50	50	57	3	12	225.8	161.98	-28.26	1.33	1.06	784.09	43.15	-94.5
Avg	-	-	-	-	-	-	-31.77%	1.33	1.04	-	-	-94.18%

7. 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, a heuristic algorithm is also proposed which is 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", *In Proceedings of DAC*, June 2002.

[2] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm", *IEEE Computer*, pp. 70-78, January 2002.

[3] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, et al., "The RAW Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs", *IEEE Micro*, pages 25-35, 2002.

[4] A. Jalabert, S. Murali, L. Benini, et al., "xpipesCompiler: A tool for instantiating application specific Networks on Chip", *DATE*, 2004.

[5] S. Murali, P. Meloni, et al., "Designing Application-Specific Networks on Chips with Floorplan Information", *ICCAD*, 2006.

[6] S. Yan and B. Lin, "Application-specific Network-on-Chip architecture synthesis based on set partitions and Steiner Trees", *ASPDAC*, 2008.

[7] K. Srinivasan, K. S. Chatha and G. Konjevod, "Linear programming based techniques for synthesis of network-on-chip architectures", *IEEE Trans. on VLSI*, 2006.

[8] S. Murali, C. Seiculescu, L. Benini, G. De Micheli, "Synthesis of Networks on Chips for 3D Systems on Chips", *ASPDAC*, 2009.

[9] C. Seiculescu, S. Murali, et al., "SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chips", *DATE*, 2009.

[10] B. Yu, S. Dong, S. Chen, S. Goto, "Floorplanning and Topology Generation for Application-Specific Network-on-Chip", *ASP-DAC*, 2010.

[11] X. Hong, S. Dong, "Non-slicing floorplan and placement using corner block list topological representation", *IEEE Trans. on CAS*, 2004.

[12] A. Pinto, et al., "Efficient Synthesis of Networks on Chip", *ICCD*, 2003.

[13] S. Chen, T. Yoshimura, "Fixed-Outline Floorplanning: Block-Position Enumeration and a New Method for Calculating Area Costs", *IEEE Trans. On CAD*, 2008.

[14] *Cbc* ILP solver, *http://projects.coin-or.org/Cbc*.

[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications. *Prentice Hall/Pearson*, 2005.

[16] A. Hansson, et al., "A Unified Approach to Constrained Mapping and Routing on Network-on-Chip Architectures", *Proc. CODES-ISSS*, 2005.

[17] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partitioning: Application in VLSI Domain", *DAC*, 1997.

[18] H. Wang, X. Zhu, L. Peh, et al., "Orion: A Power-Performance Simulator for Interconnection Networks", *Int. Symp. on Microarchitecture*, 2002.

[19] D. Bertozzi, et al., "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip", *IEEE Trans. on Parallel and Distributed Systems*, 2005.

[20] K. Srinivasan, K. Chatha, et al., "Linear programming based techniques for synthesis of network-on-chip architectures", *IEEE Trans. on VLSI*, 2006.