# Performance- and Cost-Aware Topology Generation Based on Clustering for Application-Specific Network on Chip

Fen Ge, Ning Wu, Xiaolin Qin and Ying Zhang

*Abstract*—**A clustering-based topology generation approach is proposed to construct Network-on-Chip (NoC) topologies for given applications. By exploiting communication requirements of the given application and characteristics of the router architectures, the approach constructs custom irregular NoC topology in four phases with design constraints specific to the application. Specially, a recursion based link construction algorithm embedded in the topology generation is proposed to construct links between routers. The proposed approach together with the recursive algorithm significantly reduces the network communication latency, power consumption, routers area and resource costs. The evaluation performed on various multimedia benchmark applications confirms the efficiency of the proposed approach. Experimental results show that the approach saves about 60% of the communication latency, power consumption and router area as compared to those using regular Mesh topology. Significant network resource improvement is also achieved. Moreover, the approach performs well for two multimedia applications compared to existing algorithms.**

*Index Terms*—**Cluster, network on chip, recursive link construction algorithm, topology generation**

## I. Introduction

THE rapid advancement of semiconductor technologies makes it possible to integrate dozens of cores on a single chip. With more and more cores, the on-chip communication architecture design encounters more challenges in various aspects including the throughput, latency, power consumption, signal integrity, and clock synchronization. Traditional bus-based interconnect architectures are inherently non-scalable, which constitutes a bottleneck for the on-chip communication. The emerging Network on Chip (NoC) provides an effective, reliable and flexible infrastructure for system modules based on data packet transmission scheme. It has become an effective solution to overcome difficulties associated with global interconnections and communications in complex System on Chip (SoC) designs [1].

NoC architectures are constructed using topologies. A topology describes the overall connection forms between routers and resource nodes. The floorplan of a topology determines the length and complexity of the on-chip connections, and as a result, significantly affects the network latency, throughput, area cost and power consumption. Network topologies of NoC can be classified into two categories, regular and irregular architectures. Regular topologies, as used in most NoC designs (e.g., mesh and torus), have the advantage of reusability and low design complexity. However, with regular topologies, applications cannot be well optimized. This may lead to large-scale redundant routers, low link utilization rate, and local congestion. For example, the number of routers on a mesh architecture is fixed irrespective of how many of them are actually used. The same happens to the links between routers. Even if unused routers and links can be shut down, they still occupy area on the chip. Irregular topologies, on the other hand, are designed to be application specific and therefore, are tailorable for each design. Compared to regular topologies, they usually use fewer routers and links, while offering better system performance and lower cost [2].

In this paper, we focus on network topology generation for the custom irregular architecture. Specifically, we propose a clustering-based topology generation approach for application-specific NoC. Parts of our work have been presented in [3] to minimize the network communication power consumption and resource costs. This paper expands the previous work with a further analysis of the impact of topology generation on network communication latency and area cost.

The rest of the paper is organized as follows: section II summarizes related work; section III describes the problem formulation and definitions; section IV presents our topology generation approach with an example; experimental results are discussed in section V, and finally the conclusion is made in section VI.

## II. Related Work

Regular topologies are compared and evaluated with respect to network throughput, average latency, power consumption and area in [4]. Different algorithms, such as EPAM [5], NMAP [6] and MOCA [7], are proposed to map arbitrary applications onto these regular topologies and determine the routing path between the communication core

Manuscript received January 29, 2012. This work was supported by the Natural Science Foundation of China under Grant 61076019 and 61106018, the Aeronautical Science Foundation of China under Grant 20115552031, the China Postdoctoral Science Foundation under Grant 20100481134, and the NUAA Scientific Research Foundation for talent introduction.

F. Ge, N. Wu and Y. Zhang are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjng 210016, P.R.China (e-mail: gefen@nuaa.edu.cn; wunee@nuaa.edu.cn; tracy403@nuaa.edu.cn).

X. Qin are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjng 210016, P.R.China (e-mail: qinxcs@nuaa.edu.cn).

pairs.

There are many advantages of using irregular topologies over regular topologies for application-specific NoC [8]. However, generation of irregular topologies calls for scalable topology generation algorithms [9]-[15]. In [9], the authors present a technique for constraint driven communication architecture synthesis of point to point links. The technique results in network topologies that have only two routers between each source and sink, and does not address routing for each communication trace. The work in [10] presents the mixed integer linear programming (MILP) based topology generation. However, this method is constrained by the exponentially increasing solution times for large communication trace graphs. Different optimization techniques have been proposed to address the problem of topology generation within reasonable time [11]-[14]. In [11] and [12], genetic algorithm based topology generation approaches are proposed, which obtain better results and less runtimes compared to the MILP technique. The author of [14] proposes a combination of the depth first search and the AO* algorithm to generated a near-optimal topology. However, these techniques have greater computational complexity due to a sufficient number of iterations.

In [15], a three-step topology generation algorithm called PATC is presented, which includes core cluster, core cluster optimizing and physical router mapping. The author of [16] proposes another simpler method called TopGen to cluster the given application based on the communication characteristic, and thereafter, construct the topology by connecting the clusters to each other one by one.

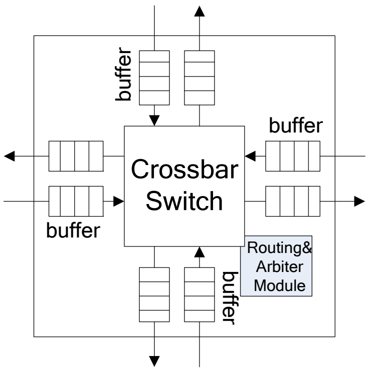
In this paper, we propose a four-phase approach of topology generation analogous to those used in [15] and [16], but completely different in the algorithm design. The proposed approach is tested and compared to those using regular NoC topology and existing algorithms on multimedia benchmarks, which shows that our approach achieves better results.

## III. Problem Formulation and Definitions

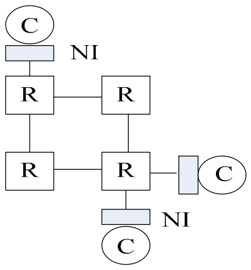
An NoC architecture consists of interconnected routers that are responsible for routing data packets on the communication architecture. As shown in Fig. 1a, a router is composed of switch fabrics, a routing and arbiter unit, an input port and output port module. Every resource node (IP core) should be connected to a router through input and output port channels, which consist of two unidirectional links. Each link can connect to a core by a network interface (NI) implemented with open core protocol (OCP), or connect to other routers directly to expand the architecture [15], as shown in Fig. 1b. In this case, designers can construct different regular or irregular NoC topologies based on the requirements and design constraints.

The topology generation problem can be formulated as follows.

**Given** a core communication graph denoted by $CCG(C, A)$, where each vertex $c_i \in C$ represents an IP core, and each directed edge $a_{i,j} \in A$ represents the communication trace from IP $c_i$ to IP $c_j$. Every edge has two attributes, denoted by $b(a_{i,j})$ and $l(a_{i,j})$, which represent the bandwidth requirement



(a) The router structure of NoC



(b) NoC architecture

Fig. 1. The router structure and NoC architecture

in bits per second (Mbps) and the latency constraint in hops respectively.

**Given** a characterized library £ of the router architectures, with η denoting the number of input and output ports of the router, and Ω denotes denoting the peak bandwidth that can be supported by the router on any one port.

**Find** a NoC topology $T(R, E)$, where $R \in$£ represents the set of routers chosen to use from library £ in the topology generation, and $E$ represents the set of links between the routers.

**Such that**:

(1) Each IP core $c$ can be mapped onto a port of a router $r$ by finding a mapping function $Map(c)$, and the maximum number of cores mapped on a router should less than η, i.e.,

$$Map: C \rightarrow R \Rightarrow r_i = Map(c_i), \forall c_i \in C, \exists r_i \in R$$

$$\forall r_k \in R, \sum_{l=0}^{\eta-1} \sum_{\forall c_i \in C} NR_{i,k,l} < \eta \tag{1}$$

where $NR_{i,k,l}$ is a {0,1} variable. If core $c_i$ is mapped onto port $p_{k,l}$ of router $r_k$, $NR_{i,k,l}=1$, otherwise, $NR_{i,k,l}=0$.

(2) For each $a_{i,j} \in A$, there exists a unique path $p_{i,j}=\{(r_i, r_k), (r_k, r_m), \dots (r_n, r_j)\} \in P$ in $T$, which satisfies the latency constraint:

$$\forall a_{i,j} \in A, \; l(a_{i,j}) \geq d(p_{Map(c_i),Map(c_j)}) \tag{2}$$

where $(r_i, r_k)$ represents the router connection $e_{i,k}$ generated by input/output ports between router $r_i$ and $r_k$, and constructs physical link $l_k$; $d(p_{i,j})$ represents the routing distance between router $r_i$ and $r_j$ along the path $p_{i,j}$.

The path $p_{i,j}$ also should satisfy the bandwidth constraint on any port of the router, i.e.,

$$\sum_{\forall a_{i,j} \in A} b(a_{i,j}) f(l_k, p_{Map(c_i),Map(c_j)}) \leq \Omega \tag{3}$$

$$f(l_k, p_{Map(c_i),Map(c_j)}) = \begin{cases} 0 : l_k \notin p_{Map(c_i),Map(c_j)} \\ 1 : l_k \in p_{Map(c_i),Map(c_j)} \end{cases}$$

Note that equation (2) means the communication traffic (workload) of the link constructed by router ports cannot exceed the supported bandwidth Ω of any one port.

(3) The total communication power consumption is minimized:

$$\min E(A) = \sum_{\forall a_{i,j} \in A} b(a_{i,j}) \times E_{bit}^{c_i,c_j} \qquad (4)$$

where

$$E_{bit}^{c_i,c_j} = \sum_{r \in p_{Map(c_i),Map(c_j)}} E_{Rbit} + \sum_{e \in p_{Map(c_i),Map(c_j)}} E_{Lbit}$$
$$= (d(p_{Map(c_i),Map(c_j)}) + 1) \times E_{Rbit} + d(p_{Map(c_i),Map(c_j)}) \times E_{Lbit}$$

$E_{bit}^{c_i,c_j}$ represents the energy consumed when one bit of data is transported through the routing path $p_{Map(c_i),Map(c_j)}$ ; $E_{Rbit}$ and $E_{Lbit}$ are the energy consumed on the router and the link respectively [15].

Since $E_{Rbit}$ and $E_{Lbit}$ are constants, the NoC power consumption varies linearly with the communication amount and routing distance, which can be represented by:

$$\min E(A) = \sum_{\forall a_{i,j} \in A} b(a_{i,j}) \times d(p_{Map(c_i),Map(c_j)}) \qquad (5)$$

(4) The network communication latency is minimized.

The traffic from any source core is subject to three types of latency in its routing path to the destination core. These are the latency through routers, the propagation latency along the links, and the packetization/de-packetization latency through NIs [17]. Since the latency through NIs is the same for all generated topologies, it is not a metric of comparison. In the average zero-load latency mode, which assumes a uncongested network condition, routers and links latency are expressed in terms of the average internode distance μ [8]. The model is a fast and efficient one to check the effect of different topologies on NoC latency [17]. Therefore, we can use the average internode distance as a metric to evaluate the network communication latency. Mathematically, the average internode distance, in hops, could be expressed as [18]:

$$\mu = \frac{\sum_{\forall a_{i,j} \in A} b(a_{i,j}) \times d(p_{Map(c_i),Map(c_j)})}{\sum_{\forall a_{i,j} \in A} b(a_{i,j})} \qquad (6)$$

The above equation (5) and (6) indicate that minimizing the average internode distance yields the same results as minimizing the communication power consumption and network latency.

Therefore, we try to cluster high communicative cores into the same router so that data exchanges among these cores consume minimized communication power consumption and network latency as calculated by (5) and (6) respectively.

(5) The network resource costs and the NoC area are minimized.

Since the final generated NoC topology *T* consists of routers and links, the network resource costs mainly determined by the number of used routers and links.

TABLE I
ROUTER AREA WITH DIFFERENT NUMBER OF PORTS

| Area | Number of Ports | | | |
|---|---|---|---|---|
| | 2 | 3 | 4 | 5 |
| μm$^2$ | 50,200 | 66,800 | 83,400 | 100,000 |

The NoC area is found to be mainly determined by router area [18], whereas the major percentage of any router area is constituted by port modules with buffers [19]. Therefore, the less the number of ports for the routers used in generated topology, the lower the NoC area. Table I shows the area of input-queue router with different number of ports, including the local port [18]. All the ports have the same buffer size and each consists of two channels for packets reception and transmission.

## IV. TOPOLOGY GENERATION APPROACH

The main idea of our proposed approach is to assign high communicative cores to the same routers or nearby routers, and subsequently, determine the optimal connection between routers. The goal is to minimize the total number of communication hops for communication IP core pairs, as well as to reduce the number of used routers and links in the NoC topology. The approach consists of four phases: 1) core clustering, 2) cluster and router mapping, 3) router connection construction, and 4) topology optimization. Each phase of the approach is described in detail as follows.

### A. Core Clustering

In the first phase, we partition the IP core set for a given application into several clusters under the design constraints. The flowchart of the clustering algorithm is shown in Fig. 2.

Step 1: Algorithm Preparation. We define a variable $N_{max}$, which denotes the maximum number of cores in each cluster. Since IP cores in the same cluster will be mapped to different ports of the same router in a topology, and each router must

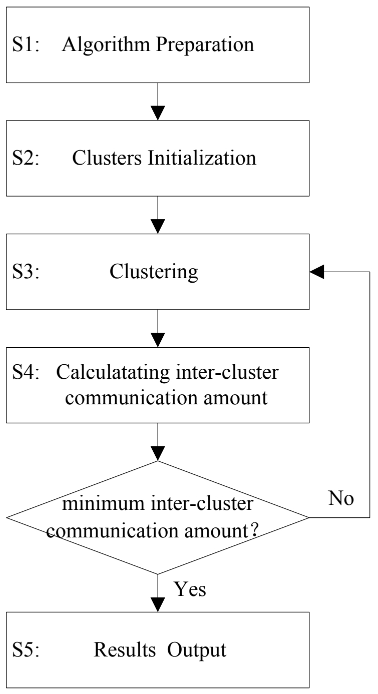
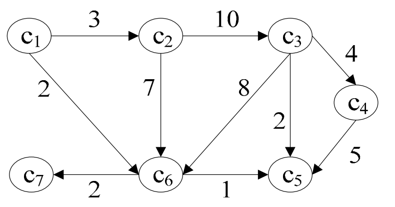
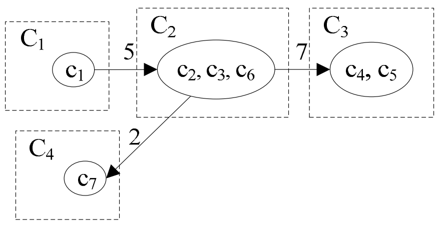

S1: Algorithm Preparation
S2: Clusters Initialization
S3: Clustering
S4: Calculatating inter-cluster communication amount
minimum inter-cluster communication amount?
No
Yes
S5: Results Output

Fig. 2. The flowchart of the clustering algorithm



(a) Core communication graph



(b) Clustering result

Fig. 3. Core clustering example

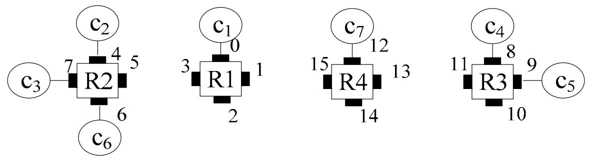


Fig. 4. Cluster and router mapping

be connected to the topology on at least one port, $N_{max}$ =η-1. Then, we sort each communication trace $a_{i,j}$ in descending order according to the communication weight $b(a_{i,j})$.

Step 2: Clusters Initialization. Clustering is to partition vertices of $CCG(C, A)$ into $k$ non-empty sets C$_1$, C$_2$, ..., C$_k$. Each cluster C$_i$ ($i$ = 1, 2 , ..., $k$) contains $N_{max}$ cores at most. In the initialization, each vertex of $CCG(C, A)$ forms a cluster partition, that is CP={C$_1$, C$_2$, ..., C$_n$}, where C$_i$= {c$_i$}, $i$ = 1, 2 , ..., $N$, $N$ is the number of vertices of $CCG$.

Step 3&4: Clusters Merging. According to the order of communication traces in step 1, we first process the edge $a_{i,j}$ with highest communication weight. Let $a_{i,j}$=($c_i$, $c_j$), if $c_i$ and $c_j$ belong to different clusters, and if the core number in the new cluster is not greater than $N_{max}$ after merging, calculate the inter-cluster communication amount among clusters after merging. If the calculated amount is less than the previous one, merge the clusters, otherwise not.

Step 5: Results Output. When all the edges have been processed in sequence, we obtain the best number of clusters with minimum inter-cluster communication amount.

For example, we give a core communication graph $CCG$ in Fig. 3a, in which the labels of the edges in $CCG$ denote the bandwidth requirement. Assuming the number of router ports η is 4, each partitioned cluster contains $N_{max}$ =4-1=3 cores at most. According to the above clustering algorithm, the $CCG$ can be divided into four clusters C$_1$, C$_2$, C$_3$, C$_4$, as shown in Fig. 3b.

## *B. Cluster and Router Mapping*

In the second phase, we map each cluster to a router. The router number used in the generated topology is equal to the number of clusters. Every IP core in the cluster is mapped to a port of a router randomly.

For the core clustering results shown in Fig. 3b, the clusters need to be mapped to four routers, denoted by $r_1$, $r_2$, $r_3$, $r_4$ respectively. As shown in Fig. 4, the core $c_1$ in the cluster C$_1$ is mapped to port 0 in the router $r_1$, and the cores in the cluster C$_2$ are mapped to three ports in the router $r_2$.

## *C. Router Connection Construction*

In the third phase, the routers mapped with IP cores are connected to form the initial topology. We sort the clusters in ascending order according to their number of cores. For clusters with the same number of cores, we sort them in descending order according to their communication amount. Then, we use a recursion based link construction algorithm to generate router connections.

Before describing the recursion based link construction algorithm, it is worth pointing out that, the communication amount of a certain cluster is calculated as the sum of the inter-cluster communication amounts between this cluster and all others. Such sort will make the communication trace with high communication weight get shortest communication path in advance, and as a result, minimize the communication power consumption.

The idea of our proposed recursion based link construction algorithm is as follows. First, the source and destination routers for each communication trace are obtained according to current router selection and port mapping results; then, under the bandwidth and latency constraints, the following three ways are attempted to recursively search the path from the source router to the destination router:

(1) Use the existing links between source and destination routers;

(2) Use the empty port of routers without placing IP core between the source and destination router to build new links;

(3) Use the links built by previous communication trace from the source or destination router to other routers.

Through the above recursively search process, we can construct router connections by allocating a routing path for each communication trace.

The pseudo code of the recursion based link construction algorithm is shown in Fig. 5. The return value of the routine $get\_next\_rtr(r_i)$ is $r_{next}$ which is connected to the router $r_i$. The constructed link between router $r_i$ and $r_{next}$ should satisfy the bandwidth and latency constraints. The adjacency matrix $RAdj$[M$_R$][M$_R$] represents the interconnection relation among routers, where M$_R$ is the number of used routers in the topology generation. The initial value of the matrix elements is 0, and the value is between 0 and ∞ if there exists a link among routers. After allocating paths for all the communication traces, each element in $RAdj$[M$_R$][ M$_R$] is checked to ensure that its value does not exceed the supported bandwidth Ω. The port information list $PortList$ is used to record the status of each router port. The status indicates whether the port is empty or connected with IP cores or other routers.

```
Algorithm Input: the corresponding source router $r_{src}$ and destination router $r_{dest}$ for each communication trace $a_{i,j}=(c_i, c_j)$.
Algorithm Output: the routing path $p_{rsrc, rdest}=\{(r_{src}, r_{next}), \cdots (r_n, r_{dest})\}$.
Recursive terminative condition: if $r_{src}==r_{dest}$ or find no path for the
  communication trace.
Recursive function: route_construction($r_{src}$, $r_{dest}$)
{ if ($r_{src}==r_{dest}$) exit;
 if (no link between $r_{src}$ and $r_{dest}$)
   { if (existing empty port $port_i$ and $port_j$ in $r_{src}$ and $r_{dest}$)
     {construct link between $port_i$ and $port_j$, let $r_{src}=r_{dest}$, add $b(a_{i,j})$ to
       $RAdj[r_{src}][r_{dest}]$, and update PortList, exit;}
    else if (no empty port in $r_{src}$)
    { $r_{next}$ = get_next_rtr($r_{src}$);
     if ($r_{next}$ != NULL)
      { let $r_{src}=r_{next}$, add $b(a_{i,j})$ to $RAdj[r_{src}][r_{next}]$;
        route_construction($r_{src}$, $r_{dest}$);}
     else add $a_{i,j}$ to PathUnAssignedSet, exit;}
    else if (no empty port in $r_{dest}$)
    { $r_{next}$ = get_next_rtr($r_{dest}$);
     if ($r_{next}$ != NULL)
      { let $r_{dest}=r_{next}$, add $b(a_{i,j})$ to $RAdj[r_{next}][r_{dest}]$;
        route_construction($r_{src}$, $r_{dest}$);}
    else add $a_{i,j}$ to PathUnAssignedSet, exit;};}
else if (existing link between $r_{src}$ and $r_{dest}$)
   { if ($RAdj[r_{src}][r_{dest}]+b(a_{i,j}) \leq \Omega$ && $d(p_{rsrc, rdest}) \leq l(a_{i,j})$)
      { let $r_{src}=r_{dest}$ , add $b(a_{i,j})$ to $RAdj[r_{src}][r_{dest}]$, exit;}
  else {
     $r_{next}$ = get_next_rtr($r_{src}$);
      if ($r_{next}$ != NULL && $r_{next}$ != $r_{dest}$)
       { let $r_{src}=r_{next}$, add $b(a_{i,j})$ to $RAdj[r_{src}][r_{next}]$;
       route_construction($r_{src}$, $r_{dest}$);}
      else if (existing empty port $port_i$ and $port_{next}$ in $r_{src}$ and $r_{next}$)
         { construct link between $port_i$ and $port_{next}$;
        let $r_{src}=r_{next}$, add $b(a_{i,j})$ to $RAdj[r_{src}][r_{next}]$;
         update PortList;
         route_construction($r_{src}$, $r_{dest}$);}
      else add $a_{i,j}$ to PathUnAssignedSet, exit;};}
}
```

Fig. 5. The pseudo code of the link construction algorithm

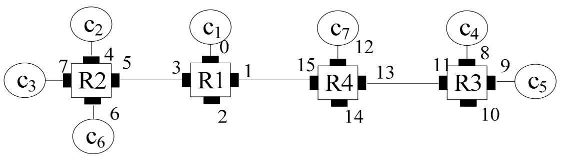


Fig. 6. Initial topology

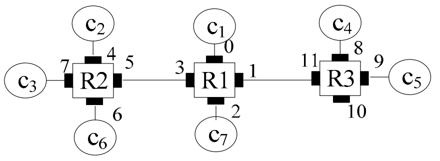


Fig. 7. Final topology

As an example, the number of cores in cluster $C_1$ and $C_4$ is identical as shown in Fig.3b, and the communication amount of cluster $C_1$ is 5 which is larger than that of cluster $C_4$. As a result, the routing path for communication trace between cluster $C_1$ and $C_2$ is allocated first, and port 3 is connected to port 5 to construct a routing path. Then, the routing paths for other two communication traces between $C_4$ and $C_2$, $C_3$ and $C_2$ can be allocated. Eventually, after completing path allocations for all the communication traces, connection among routers can be constructed. The initial topology of the mapping results in Fig. 4 is shown in Fig. 6.

### *D. Topology Optimization*

The last phase is to merge adjacent routers with empty ports until no adjacent routers can be merged. This further reduces communication power consumption and resources costs. As an example shown in Fig. 6, there exist empty ports in router $r_1$ and $r_4$, thus router $r_1$ can be merged with router $r_4$, leading to the final NoC topology as is shown in Fig.7.

In order to evaluate the time complexity of our proposed approach, let $n$ be the number of vertices in the core communication graph, and $a$ be the number of edges in the core communication graph *CCG*. Since each cluster contains at most $n$ elements and there exists a maximum of $n$ clusters, the complexity of inter-cluster communication amount calculation is $O(n^2)$. All the edges should be traversed, so the time complexity of cluster partitioning is $O(a\times n^2)$. Consequently, the overall time complexity of the algorithm is estimated to be $O(a\times n^2)$.

## V. Experimental Results

In this section, we present the experimental results obtained by executing the proposed approach on various multimedia benchmark applications. We generated custom irregular NoC topologies for seven combinations of four multimedia benchmarks: MP3 audio encoder, MP3 audio decoder, H.263 video encoder, and H.263 video decoder [5]. In addition, we obtained results for three other benchmarks: MPEG4 decoder, video object plane decoder (VOPD), and multi-window display (MWD) [2]. Table II lists the graph IDs and sizes of the *CCG* of the various benchmarks.

In order to evaluate the efficiency of the proposed approach, we compared the results produced by our clustering-based topology generation approach (Cluster-TG) against the solution of mapping benchmark applications onto regular Mesh topology. The selection of Mesh topology for comparison is due to the fact that, Mesh topology is proved to outperform other regular NoC topologies with respect to power consumption and area costs, and it can be easily implemented on chips. The number of router ports η is set to be 4, and the supported bandwidth Ω is set to be 1GB/s.

TABLE II
Graph Characteristics

| Graph | Graph ID | Nodes | Edges |
|---|---|---|---|
| MP3 decoder | G1 | 6 | 6 |
| H.263 decoder | G2 | 7 | 8 |
| MP3 encoder | G3 | 7 | 8 |
| H.263 encoder | G4 | 8 | 11 |
| MWD | G5 | 12 | 13 |
| VOPD | G6 | 12 | 15 |
| MPEG4 decoder | G7 | 12 | 26 |
| H.263 enc MP3 dec | G8 | 12 | 17 |
| H.263 enc MP3 enc | G9 | 14 | 19 |
| H.263 enc H.263 dec | G10 | 15 | 19 |

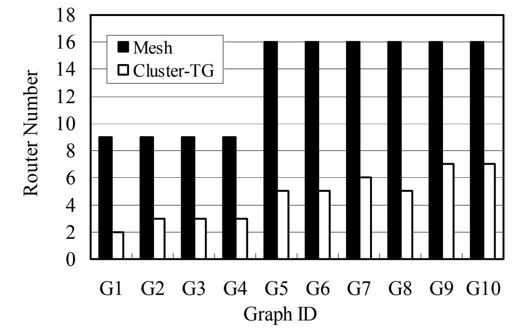
TABLE III
THE AVERAGE INTERNODE DISTANCES AND ROUTER AREA COMPARISON

| Graph ID | Average Internode Distance (hops) | | Ratio(2/1) | Router Area ($\mu m^2$) | | Ratio(4/3) |
|---|---|---|---|---|---|---|
| | Optimal-Mesh(1) | Cluster-TG(2) | | Optimal-Mesh(3) | Cluster-TG(4) | |
| G1 | 1 | 0.09 | 0.090 | 651,000 | 166,800 | 0.256 |
| G2 | 1.03 | 0.06 | 0.058 | 667,600 | 233,600 | 0.350 |
| G3 | 1.10 | 0.22 | 0.200 | 651,000 | 233,600 | 0.359 |
| G4 | 1.00 | 0.35 | 0.350 | 684,200 | 250,200 | 0.366 |
| G5 | 1.37 | 0.71 | 0.518 | 1,284,600 | 417,000 | 0.325 |
| G6 | 1.43 | 0.54 | 0.378 | 1,268,000 | 417,000 | 0.329 |
| G7 | 0.66 | 0.64 | 0.970 | 1,268,000 | 467,200 | 0.368 |
| G8 | 1.27 | 0.56 | 0.440 | 1,268,000 | 417,000 | 0.329 |
| G9 | 1.26 | 0.54 | 0.429 | 1,301,200 | 550,600 | 0.423 |
| G10 | 1.13 | 0.39 | 0.345 | 1,317,800 | 567,200 | 0.430 |

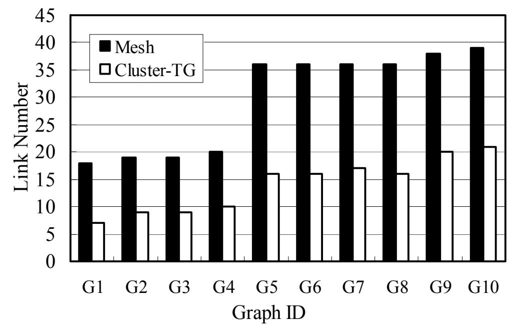
Fig.8 presents the results of the comparison in communication power consumption of NoC topology generated by Random-Mesh, Optimal-Mesh and Cluster-TG. 'Random-Mesh' represents the solution of mapping IP cores in benchmark applications onto regular Mesh topology randomly. 'Optimal-Mesh' represents the solution of mapping IP cores onto optimized regular Mesh topology by the genetic algorithm based approach in [20]. Fig. 9 shows the comparison of router and link utilities. Total router area of the generated topologies for these benchmark applications is listed in Table III.

As is seen from the figures, much better results in communication power consumption and resource costs have been achieved using our approach compared to that of the regular Mesh topology. On average, our approach saves about 61.5% of communication power consumption compared to Optimal-Mesh. The router area is reduced by about 64.6% with respect to Optimal-Mesh. As an example, the *CCG*, the optimal mapping result and the generated irregular topology of the MPEG4 decoder are illustrated in Fig.10. As shown in Fig. 10b, three 2-port, two 3-port, seven 4-port and four 5-port routers are needed in the generated topology by Optimal-Mesh. However, only two 3-port and four 4-port routers are needed by our Cluster-TG approach. Therefore, our proposed approach is expected to perform better than using regular Mesh topology from the network cost point of view.

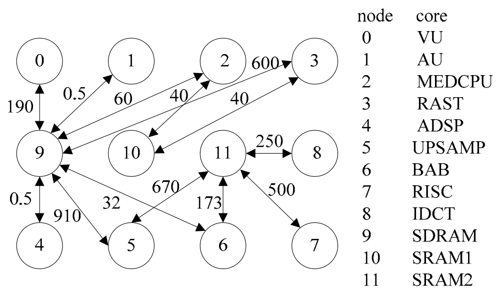
The impact of our approach on the network communication latency can be seen in Table III, where the average internode distance $\mu$ is calculated by (6). As is shown in the table, about 62.2% reduction in $\mu$ is achieved as compared to Optimal-Mesh. Therefore, our proposed approach is expected to perform better than using regular Mesh topology from the network performance point of view.



(a) The number of routers



(b) The number of links
Fig. 9. Resource costs comparison



(a) The CCG of MPEG4 decoder

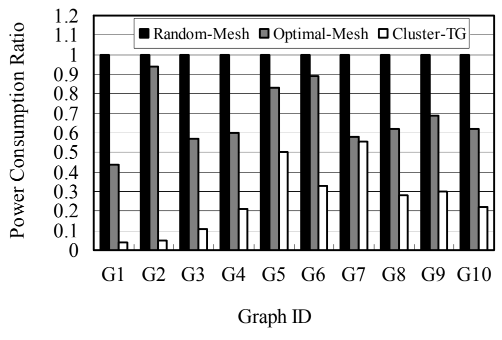


Fig. 8. Communication power consumption comparison

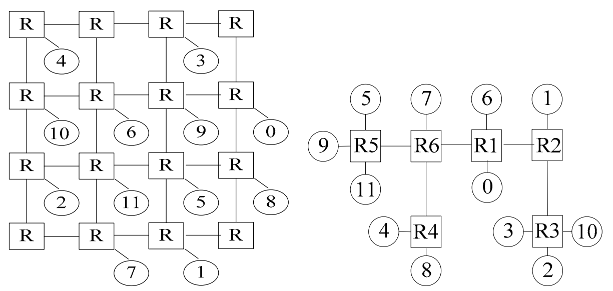

(b) The optimal mapping result (c) The generated topology by Cluster-TG
Fig. 10. The CCG and generated topologies of MPEG4 decoder

TABLE IV
THE AVERAGE INTERNODE DISTANCES AND ROUTER AREA COMPARISON FOR DIFFERENT APPROACH

| Application | Topology Generation Approach | Average Internode Distance | | Router Area | |
|---|---|---|---|---|---|
| | | hops | % Cluster-TG | μm$^2$ | % Cluster-TG |
| VOPD | Cluster-TG | 0.540 | 100 | 417,000 | 100 |
| | TopGen | 0.540 | 100 | 417,000 | 100 |
| | PATC | 0.541 | 100.2 | 417,000 | 100 |
| MWD | Cluster-TG | 0.714 | 100 | 417,000 | 100 |
| | TopGen | 0.714 | 100 | 417,000 | 100 |
| | PATC | 0.829 | 116.1 | 417,000 | 100 |

Another experiment is conducted to compare the results generated by Cluster-TG, TopGen [16] and PATC [15] in two multimedia applications, VOPD and MWD. The resource costs of the applications using different approaches turn out to be about the same, so is the router area as listed in Table IV. The power consumptions of different approaches are compared in Fig. 11, and the average internode distances are compared in Table IV. It can be seen that our proposed approach achieves results that are better than PATC and commensurate with TopGen. As an example, the *CCG*s and the generated irregular topologies of the VOPD and MWD benchmarks are illustrated in Fig. 12 and Fig. 13 respectively.

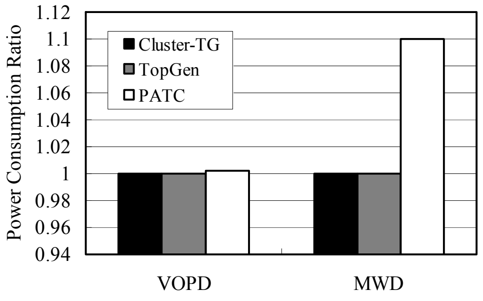
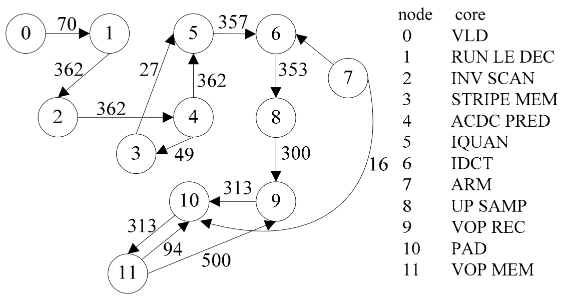
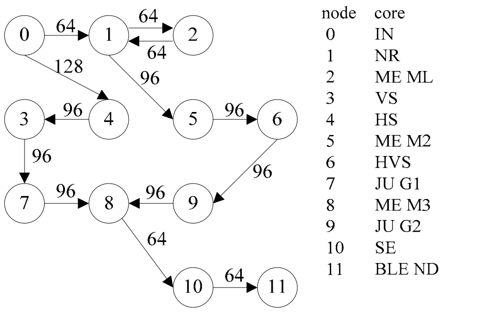


Fig.11. Power consumption comparison for different approach



| node | core |
|---|---|
| 0 | VLD |
| 1 | RUN LE DEC |
| 2 | INV SCAN |
| 3 | STRIPE MEM |
| 4 | ACDC PRED |
| 5 | IQUAN |
| 6 | IDCT |
| 7 | ARM |
| 8 | UP SAMP |
| 9 | VOP REC |
| 10 | PAD |
| 11 | VOP MEM |

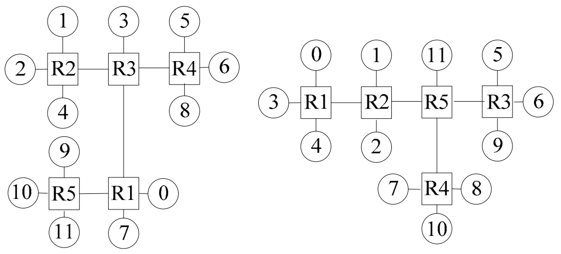
(a) The CCG of VOPD



| node | core |
|---|---|
| 0 | IN |
| 1 | NR |
| 2 | ME ML |
| 3 | VS |
| 4 | HS |
| 5 | ME M2 |
| 6 | HVS |
| 7 | JU G1 |
| 8 | ME M3 |
| 9 | JU G2 |
| 10 | SE |
| 11 | BLE ND |

(b) The CCG of MWD

Fig. 12. The CCG of application VOPD and MWD



(a) The topology of VOPD (b) The topology of MWD

Fig. 13. The final irregular topology of application VOPD and MWD

## VI. CONCLUSION AND FUTURE WORK

This paper presents a four-phase clustering-based topology generation approach for application-specific NoC. The aim is to reduce the network communication latency from the performance point of view, and the communication power consumption, router area and resource costs from the cost point of view. Under the constraints of the bandwidth and latency, the approach designs custom irregular NoC topologies according to the communication requirements of the given application and characteristics of router architectures. Specially, a recursion based link constructing algorithm embedded in the topology generation is proposed to construct links between routers. Applying our approach on various multimedia benchmark applications gives experimental results showing significantly improved performance as compared to those using regular Mesh topology and existing algorithms.

The advent and increasing viability of 3D silicon integration technology make it possible to scale NoC over the third dimension. As a result, 3D NoC is arousing more and more research interest. Future work will focus on an extension of our approach to application-specific 3D topology generation with metrics of 3D NoC taken into consideration.

## REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, vol.35, no.1, pp. 70-78, Jan. 2002.

[2] D. Bertozzi, A. Jalabert, S. Murali, et al., "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113-129, Feb. 2005.

[3] F. Ge, N. Wu, X. Qin, and Y. Zhang, "Clustering-based topology generation approach for application-apecific network on chip," in *Lecture Notes in Engineering and Computer Science*: *Proceedings of the Word Congress on Engineering and Computer Science 2011*, WCECS 2011, October 19-21, 2011, San Francisco, USA, pp. 753-757.

[4] P. P. Pande, C. Grecu, M. Jones, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Trans. on Computers*, vol. 54, no. 8, pp.

1025-1040, Aug. 2005.

[5] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Trans. on computer-aided design of integrated circuits and systems*, vol. 24, no. 4, pp. 551-562, April. 2005.

[6] S. Murali and G. De. Micheli, "Bandwidth- constrained mapping of cores onto NoC architecture," in *Proc. Design, Automation and Test in Europe Conference and Exhibition*, 2004, pp. 896-901.

[7] K. Srinivasan and K. S. Chatha, "A technique for low energy mapping and routing in network-on-chip architectures," in *Proc. Int. Sympo. Low Power Electronics and Design*, 2005, pp. 387-392.

[8] U. Ogras and R. Marculescu, "It's a small word after all: NoC performance optimization via long-rang link insertion," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 14, no. 7, pp. 693-706, Jul. 2006.

[9] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli, "Efficient synthesis of networks on chip," in *Proc. Int. Conf. Computer Design*, 2003, pp. 146-150.

[10] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear-programming-based techniques for synthesis of network-on-chip architectures," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 14, no. 4, pp. 407-420, 2006.

[11] K. Srinivasan and K. S. Chatha, "ISIS: a genetic algorithm based technique for custom on-chip interconnection network synthesis," in *Proc. Int. Conf. VLSI Design*, 2005, pp. 623-628.

[12] G. Leary, K. Srinivasan, K. Mehta, and K. S. Chatha, "Design of network on chip architectures with a genetic algorithm-based technique," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 17, no. 5, pp. 674-687, 2009.

[13] N. Choudhary, M. S. Gaur, V. Laxmi, and V. Singh, "Genetic algorithm based topology generation for application specific network-on-chip," in *Proc. IEEE Int. Sympo. Circuits and Systems (ISCAS)*, 2010, pp. 3156-3159.

[14] Z. Liu, J. Cai, L. Yao, and M. Du, "Application-aware generation and optimization for NoC topology," in *Proc. IEEE Youth Conf. Information, Computing and Telecommunication*, 2009, pp. 259-262.

[15] K. C. Chang and T. F. Chen, "Low-power algorithm for automatic topology generation for application- specific networks on chips," *IET Computers & Digital Techniques*, vol. 2, no. 3, pp. 239-249, 2008.

[16] Y. Ar, S. Tosun, and H. Kaplan, "TopGen: a new algorithm for automatic topology generation for network on chip architectures to reduce power consumption," in *Proc. AICT*, 2009, pp.1-5.

[17] V. Pavlidis and E. Friedman, "3-D topologies for networks-on-chip," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 15, no.10, pp. 1081-1090, Oct. 2007.

[18] A. A. Morgan, H. Elmiligi, M. Watheq El-Kharashi, and F. Gebali, "Network-on-chip topology generation techniques: area and delay evaluation," in *Proc. 3rd Int. Design and Test Workshop*, 2008, pp. 33-38.

[19] M. Coenen, S. Murali, A. Radulescu, K. Goossens, and G. D. Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," in *Proc. Third IEEE/ACM/IFIP Int. Conf. Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct. 22-25, 2006, Seoul, Korea, pp. 130-135.

[20] F. Ge and N. Wu, "Genetic algorithm based mapping and routing approach for network on chip architectures," *Chinese Journal of Electronics*, vol.19, no.1, pp. 91-96, Jan. 2010.