Clustering-Based Topology Generation Approach for Application-Specific Network on Chip

Fen Ge, Ning Wu, Xiaolin Qin and Ying Zhang

Abstract—**In this paper, we present a clustering-based topology generation approach to construct the Network on Chip (NoC) topology for given applications such that the network communication power consumption and resource costs are minimized. The approach consists of four phases and constructs custom irregular NoC topology with design constraints, according to the communication requirements of the given application and characteristics of the router architectures. Especially, a recursion based link construction algorithm embedded in the topology generation is proposed to construct links between routers. The evaluation performed on various multimedia benchmark applications confirms the efficiency of the proposed approach. Experimental results show that the approach saves 61.5% of power consumption on average in comparison with using regular Mesh topology, and achieves significant network resources improvement. The approach also performs well for two multimedia applications compared to existing algorithms.**

Index Terms—**Cluster, network on chip, power consumption, topology generation**

I. Introduction

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 the problems of global interconnection and communication 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, and its floorplan determines the length and complexity of the on-chip connections, which has important effects on network delay, throughput, area cost and power consumption. Network topologies of NoC can be classified into regular and irregular architectures. Regular topologies used in most NoC designs (e.g., mesh and torus) have advantages of topology reuse and low design complexity. However, applications cannot be well optimized on regular topologies, which may lead to large-scale redundant routers, lower link utilization rate, and local congestion. Irregular topologies designed to be application specific are tailorable for each design. Compared to regular topologies, they usually use fewer routers and links, as well as offer better system performance and lower cost [2]. Thus, in this paper, we focus on network topology generation for the custom irregular architecture.

Although the advantages of irregular topologies are clear enough, there are still needs for scalable topology generation algorithms [3-9]. In [3], 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 [4] presents mixed integer linear programming (MILP) based topology generation. However, this method is constrained by 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 [5-8]. In [5] and [6], genetic algorithm based topology generation approaches are proposed, which obtain better results and less runtimes compared to the MILP technique. The author of [8] proposes a combination of depth first search and AO* algorithm to generated a near-optimal topology. However, these techniques have greater computational complexity due to a sufficient number of iterations. In [9], a three-step topology generation algorithm called PATC is presented, which includes core cluster, core cluster optimizing and physical router mapping. The author of [10] proposes another simpler method called TopGen to cluster the given application based on the communication characteristic, then to construct the topology by connecting the clusters to each other one by one.

In this paper, we propose a clustering-based topology generation approach for application-specific NoC. Our approach is composed of four phases, and shows similarities with the methods in [9] and [10]. However, the algorithms we apply in these steps are completely different. We compare our approach with using regular NoC topology and existing algorithms on multimedia benchmarks and the approach can achieve better results.

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

II. Problem Formulation and Definitions

A NoC architecture consists of interconnected routers, that are responsible for routing data packets on the

Manuscript received July 16, 2011. This work was supported by the Natural Science Foundation of China under Grant 61076019, 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).

ISBN: 978-988-19251-7-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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 [9], 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 in bits per second (Mbps)and the latency constraint in hops respectively.

Given a characterized library £ of router architectures, where η denotes the number of input and output ports of the router, and Ω denotes the peak bandwidth that the router can support 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 topology generation, and E represents the set of links between routers.

Such that:

(1) Each IP core c can be mapped onto a port of a router r, and the maximum number of cores mapped on a router should less than η.

(2) For each $a_{i,j} \in A$, there exists a unique path $p_{i,j}=\{(r_i, r_k), (r_k, r_m), \ldots (r_n, r_j)\} \in P$ in T that satisfies communication latency and bandwidth constraints.

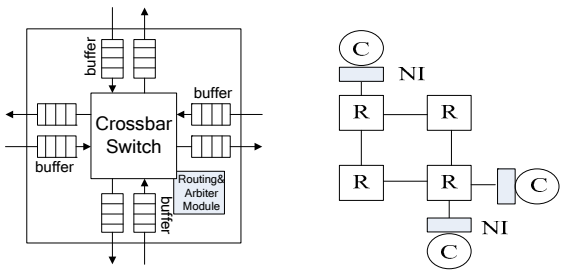
(3) The total communication power consumption and network resource costs (the number of used routers and links) are minimized.

In uncongested network conditions, the NoC power consumption varies linearly with the communication amount and routing distance, which can be represented by:

$$\min E(A) = \sum_{\forall a_{i,j} \in A} b(a_{i,j}) \times d(p_{i,j}) \qquad (1)$$

where $d(p_{i,j})$ represents the routing distance between router r_i and r_j along the path $p_{i,j}$.

Therefore, we will try to cluster high communicative cores into the same router so that data exchanges among these cores consume minimized power consumption as calculated by (1).



（a）The router structure of NoC （b）NoC architecture

Fig. 1. The router structure and NoC architecture

III. 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 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 optimizing. Each phase of the approach is described in detail as follows.

A. Core Clustering

In the first phase, we partition the IP cores set for a given application into several clusters under the design constraints. The flow 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 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 $C_1, C_2, \ldots, C_k$. Each cluster C_i ($i = 1, 2, \ldots, k$) contains N_{max} cores at most. In the initialization, each vertex of $CCG(C, A)$ forms a cluster partition, that is $CP=\{C_1, C_2, \ldots, C_n\}$, where $C_i = \{c_i\}$, $i = 1, 2, \ldots, N$, 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 the core number in the new cluster is not greater than N_{max} after merging, then calculate the inter-cluster communication amount among clusters after merging. If the calculated amount is less than that of the previous, merge the clusters, otherwise not.

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

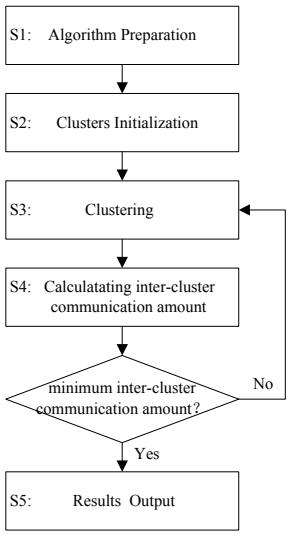
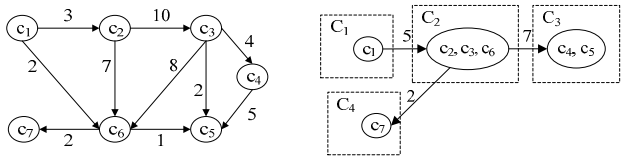


Fig. 2. The flow of the clustering algorithm

ISBN: 978-988-19251-7-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



(a) Core communication graph (b) Clustering result

Fig. 3. Core clustering example

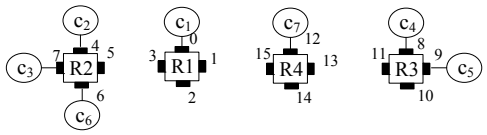


Fig. 4. Cluster and router mapping

For example, we give a core communication graph *CCG* in Fig. 3a, and 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. If the number of cores is the same, 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 cluster communication amount is the summation of inter-cluster communication amount between this cluster and 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 delay 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 constraints of the bandwidth and latency. 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) ≤Ω && d(p_rsrc, rdest) ≤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

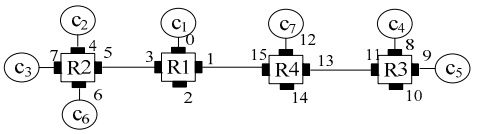


Fig. 6. Initial topology

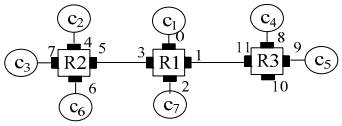


Fig. 7. Final topology

As an example, the number of cores in cluster C_1 and C_4 is the same as shown in Fig.3b, and the communication amount of cluster C_1 is 5 which are 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 firstly, 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 Optimizing

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 exists 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.

E. Algorithm Complexity Analysis

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 as $O(a \times n^2)$.

IV. 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 [11]. In addition we obtained results for three other benchmarks: MPEG4 decoder, video object plane decoder (VOPD), and multi-window display (MWD) [2]. Table I 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 generated by our clustering-based topology generation approach (Cluster-TG) against solution of mapping benchmark applications onto regular Mesh topology. The number of router ports η is set to be 4, and the supported bandwidth Ω is set to be 1GB/s.

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 [12]. Fig. 9 shows the comparison of router and link utilities. As seen from the figures, a much better performance in communication power consumption and resource costs has 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.

TABLE I
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

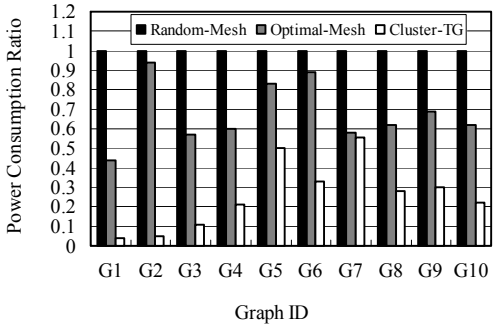
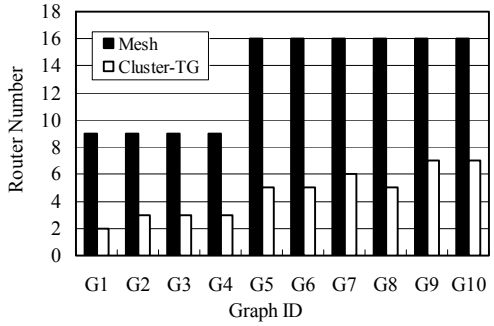
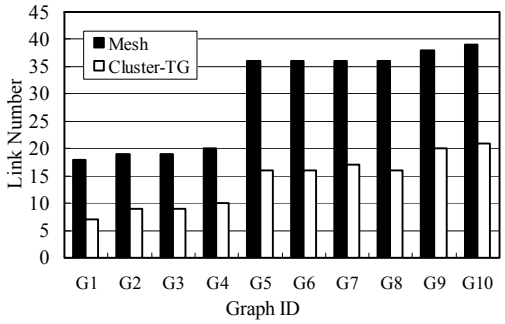


Fig. 8. Communication power consumption comparison



(a) The number of routers

ISBN: 978-988-19251-7-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



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

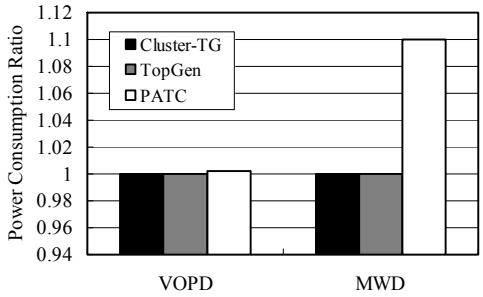
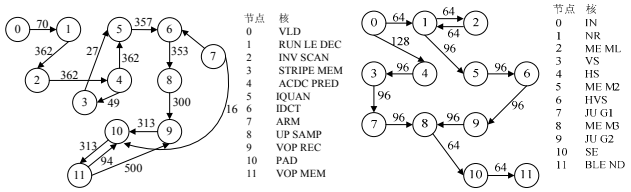
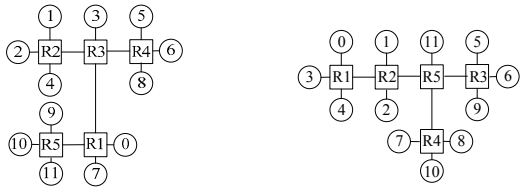


Fig.10. Power consumption comparison for different approach



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

Fig. 11. The CCG of application VOPD and MWD



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

Fig. 12. The final irregular topology of application VOPD and MWD

Another experiment is conducted to compare the results of two multimedia applications, VOPD and MWD, generated by Cluster-TG, TopGen [10] and PATC [9] respectively. The resource costs of the applications using different approaches turn out to be about the same, and the power consumptions are compared in Fig. 10. 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. 11 and Fig. 12 respectively.

V. Conclusion

This paper presents a four phase clustering-based topology generation approach for application-specific NoC. The aim is to reduce the network communication power consumption and resource costs. 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. Especially, a recursion based link constructing algorithm embedded in the topology generation is proposed to construct links between routers. We applied our approach on various multimedia benchmark applications with experimental results showing improved performance as compared with using regular Mesh topology and existing algorithms.

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] 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.

[4] K. Srinivasan, K. S. Chatha and Konjevod G, "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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] J. Hu, R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in Proc. *ACM/IEEE ASP-DAC*, 2003, pp. 233-239.

[12] 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.

ISBN: 978-988-19251-7-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)