Chapter 11
Power-Aware Topology Generation Based on Clustering for Application-Specific Network on Chip

Fen Ge and Ning Wu

Abstract A clustering-based topology generation approach is proposed to construct Network on Chip (NoC) topologies for given applications. The approach consists of four phases and constructs irregular NoC topology with design constraints, according to the communication requirements of the given application and characteristics of the router architectures. Specially, 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. Significant network resource improvement is also achieved. Moreover, the approach performs well for two multimedia applications compared to existing algorithms.

Keywords 3D · Application-specific · Cluster · Network on chip · Power consumption · Topology generation

F. Ge (✉) · N. Wu
College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, 210016 Nanjng, P.R.China
e-mail: gefen@nuaa.edu.cn

N. Wu
e-mail: wunee@nuaa.edu.cn

11.1 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 chapter, 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. This chapter expands the previous work with a further analysis of the feasibility to address the problem of application-specific 3D NoC topology generation using the proposed approach.

The rest of the chapter is organized as follows: Section 11.2 summarizes related work; Sect. 11.3 describes the problem definitions; Sect. 11.4 presents our topology generation approach with an example; Sect. 11.5 discusses the possible extension of the current approach; experimental results are discussed in Sect. 11.6, and finally the conclusion is made in Sect. 11.7.

11.2 Related Work

There are many advantages of using irregular topologies over regular topologies for application-specific NoC [4]. However, generation of irregular topologies calls for scalable topology generation algorithms [5–11]. In [5], 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 [6] 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 [7–10]. In [7] and [8], genetic algorithm based topology generation approaches are proposed, which obtain better results and less runtimes compared to the MILP technique. The author of [10] 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 [11], a three-step topology generation algorithm called PATC is presented, which includes core cluster, core cluster optimizing and physical router mapping. The author of [12] 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 chapter, we propose a four-phase approach of topology generation analogous to those used in [11] and [12], but completely different in the algorithm design. The proposed approach is verified and compared to those using regular NoC topology and existing algorithms on multimedia benchmarks, which shows that our approach achieves better results.

11.3 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. 11.1a, 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 [11], as shown in Fig. 11.2b. In this case, designers can construct different regular or irregular NoC topologies based on the requirements and design constraints.

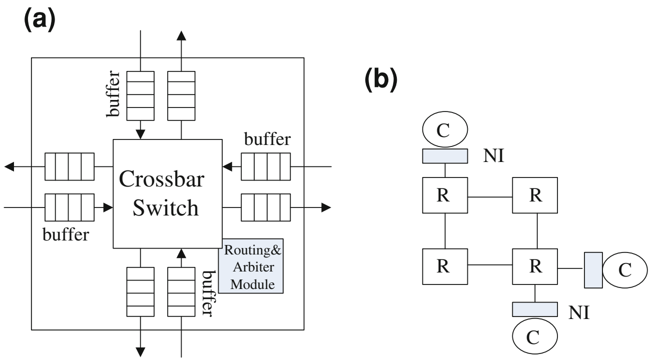


Fig. 11.1 The router structure and NoC architecture, **a** The router structure of NoC **b** NoC architecture

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 the router architectures, with η denoting the number of input and output ports of the router, and Ω denoting the peak bandwidth that can be supported by the router on any one port.

Find a NoC topology T(R, E), where R ∈ £ 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, 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 is minimized:

$$\min E(A) = \sum_{\forall a_{i,j} \in A} b(a_{i,j}) \times E_{bit}^{c_i,c_j} \tag{11.1}$$

where

$$\begin{aligned} E_{bit}^{c_i,c_j} &= \sum_{r \in p_{i,j}} E_{Rbit} + \sum_{e \in p_{i,j}} E_{Lbit} \\ &= (d(p_{i,j}) + 1) \times E_{Rbit} + d(p_{i,j}) \times E_{Lbit} \end{aligned}$$

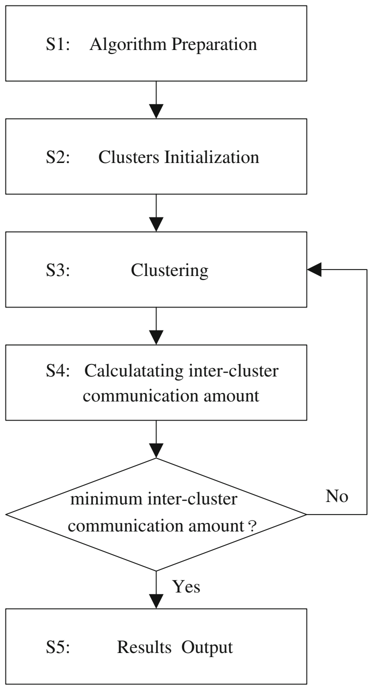

S1: Algorithm Preparation
S2: Clusters Initialization
S3: Clustering
S4: Calculatating inter-cluster communication amount
minimum inter-cluster communication amount ?
No
Yes
S5: Results Output

Fig. 11.2 The flowchart of the clustering algorithm

$E_{bit}^{c_i,c_j}$ represents the energy consumed when one bit of data is transported through the routing path $p_{i,j}$; E_{Rbit} and E_{Lbit} are the energy consumed on the router and the link respectively [11].

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:

$$\min E(A) = \sum_{\forall a_{i,j} \in A} b(a_{i,j}) \times d(p_{i,j}) \tag{11.2}$$

Therefore, we try to cluster high communicative cores into the same router so that data exchanges among these cores consume minimized communication power consumption as calculated by (11.2).

11.4 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.

11.4.1 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. 11.2.

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, $N_{max} = \eta - 1$. 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 and 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 CCG in Fig. 11.3a, 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. 11.3b.

11.4.2 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.

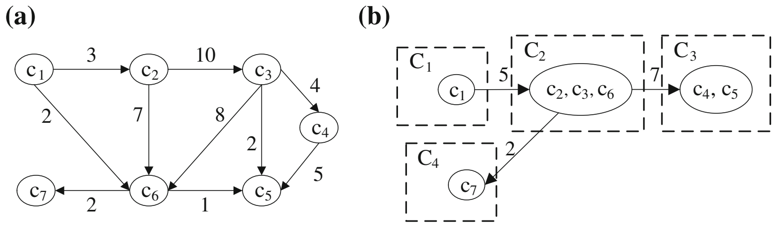
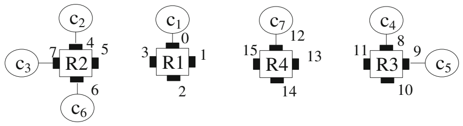


Fig. 11.3 Core clustering example, **a** Core communication graph **b** Clustering result

Fig.11.4 Cluster and router mapping



For the core clustering results shown in Fig. 11.3b, the clusters need to be mapped to four routers, denoted by r_1, r_2, r_3, r_4 respectively. As shown in Fig. 11.4, 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.

11.4.3 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. 11.5. 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*[$\mathrm{M_R}$][$\mathrm{M_R}$] represents the interconnection relation among routers, where $\mathrm{M_R}$ 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*[$\mathrm{M_R}$][$\mathrm{M_R}$] 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.

As an example, the number of cores in cluster $\mathrm{C_1}$ and $\mathrm{C_4}$ is identical as shown in Fig. 11.3b, and the communication amount of cluster $\mathrm{C_1}$ is 5 which is larger than that of cluster $\mathrm{C_4}$. As a result, the routing path for communication trace between cluster $\mathrm{C_1}$ and $\mathrm{C_2}$ 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 $\mathrm{C_4}$ and $\mathrm{C_2}$, $\mathrm{C_3}$ and $\mathrm{C_2}$ 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. 11.4 is shown in Fig. 11.6.

11.4.4 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. 11.6, 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. 11.7.

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

```
    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 $p_{rsrc, rdest}=\{(r_{src}, r_{next}), \ldots (r_n, r_{dest})\}$.
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 ($RAdj[r_{src}][ r_{dest}]+ b(a_{i,j})\leq$      && $d(p_{rsrc, rdest}) \leq l(a_{i,j})$)
      { 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. 11.5 The pseudo code of the link construction algorithm

Fig. 11.6 Initial topology

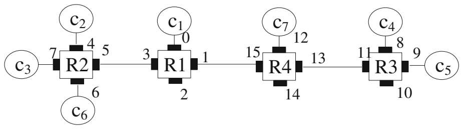


Fig. 11.7 Final topology

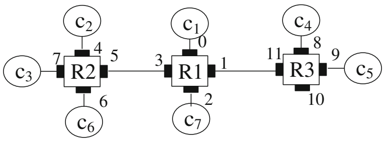


Table 11.1 Graph Characteristics

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

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

11.5 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 11.1 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,

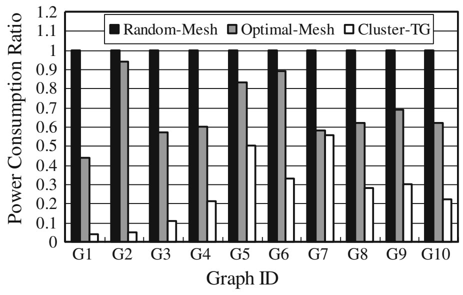
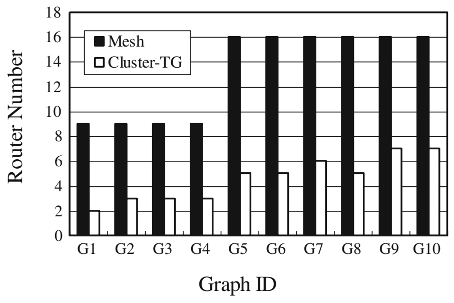
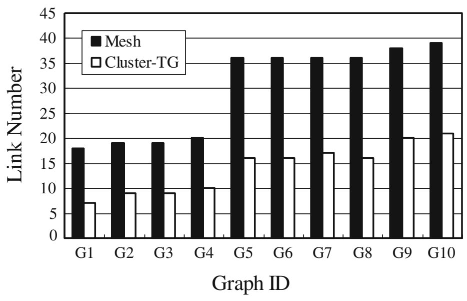


Fig.11.8 Communication power consumption comparison



(a) The number of routers



(b) The number of links

Fig.11.9 Resource costs comparison **a** The number of routers, **b** The number of links

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 1 GB/s.

Figure 11.8 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 [13]. Figure 11.9 shows the comparison of router and link utilities. As seen from the figures, a much better

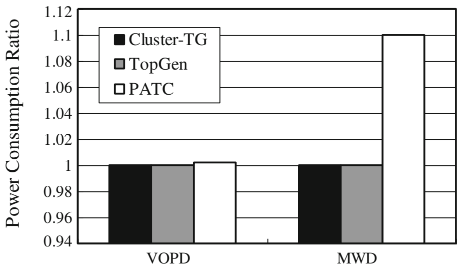
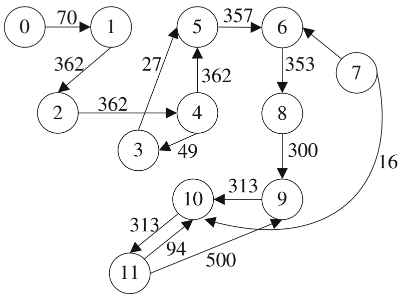
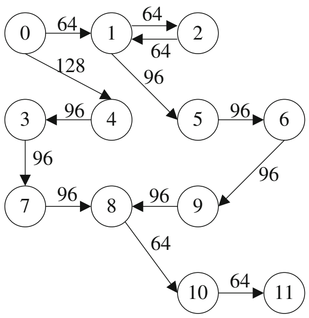


Fig. 11.10 Power consumption comparison for different approach



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. 11.11 The CCG of application VOPD and MWD, **a** The CCG of VOPD, **b** The CCG of MWD

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.

Another experiment is conducted to compare the results of two multimedia applications, VOPD and MWD, generated by Cluster-TG, TopGen [12] and PATC [11] 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. 11.10. It can be seen that our proposed approach achieves results that are

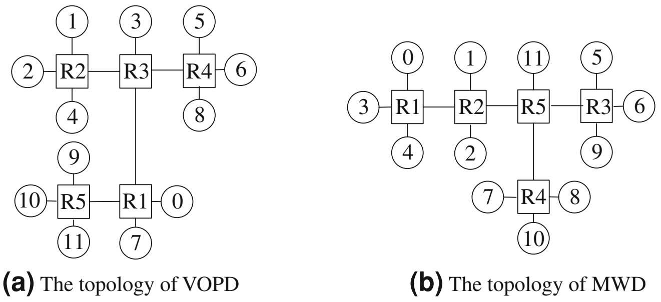


Fig. 11.12 The final irregular topology of application VOPD and MWD

better than PATC, and commensurate with TopGen. As an example, the CCGs and the generated irregular topologies of the VOPD and MWD benchmarks are illustrated in Figs. 11.11 and 11.12 respectively.

11.6 Possible Extension

The advent and increasing viability of 3D silicon integration technology make it possible to scale NoC over the third dimension [14]. As a result, 3D NoC is arousing more and more research interest. My proposed approach can be extended to application-specific 3D topology generation with metrics of 3D NoC taken into consideration.

In 3D NoC, IP cores are distributed on different 2D layers, and multiple device layers are stacked on top of each other with direct vertical interconnects tunneling through them using through-silicon vias (TSVs). Every IP core also should be connected to a router in 2D layers. The router connects to other routers in the same layer using horizontal links, and connects to other routers in the adjacent layers using up/down port and vertical links.

The approach for application-specific 3D NoC topology generation also should consist of four phases: core clustering, cluster and router mapping, router connection construction, and topology optimization. However, the problem introduces new issues, such as the technology constraint on the number of TSVs that can be supported, accurate power models for 3D interconnects.

In the phase of core clustering we first partition the IP core set for a given application (the example CCG is shown in Fig. 11.13a) into several clusters under the constraint on the number of TSVs, and make the IP cores in different clusters distribute on different 2D layers, as shown in Fig. 11.13b. Then IP cores in the same layer are further partitioned into clusters according to the algorithm in Sect. 11.4.1. In the phase of router connection construction, the routing path allocations for communication traces maybe use the vertical links among routers in

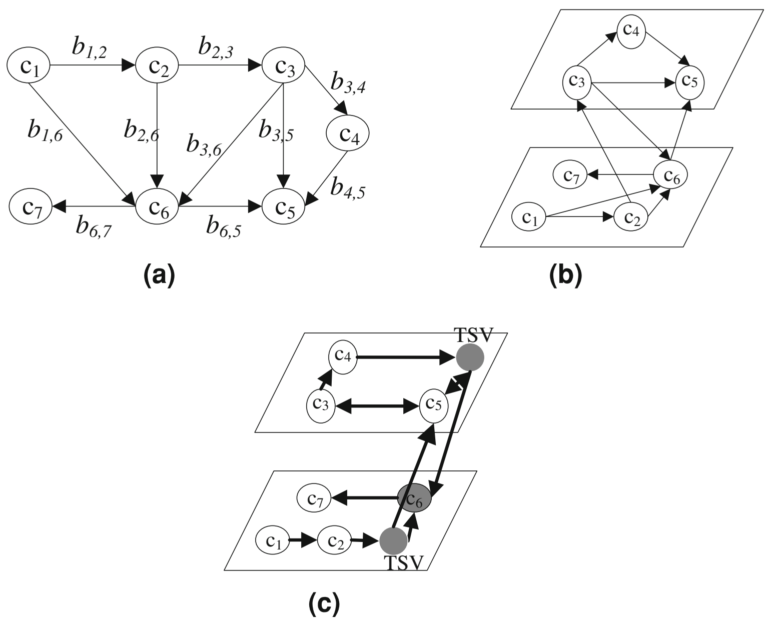


Fig. 11.13 Illustration of the 3D NoC topology generation problem, **a** Core communication graph **b** Clustering result, **c** The construction of vertical links

adjacent layers, as shown in Fig. 11.13c. The construction of vertical links should meet the constraint on the number of TSVs

Additionally, the power model in 2D NoC should be extended to 3D NoC by including the power consumed on vertical links.

11.7 Conclusion and Future Work

This chapter presents a four-phase clustering-based topology generation approach for application-specific NoC. The aim is to reduce the network communication power consumption. 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 detail analysis of 3D NoC topology generation using our approach will be done as future work.

Acknowledgments 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.

References

1. Benini L, De Micheli G (2002) Networks on chips: a new SoC paradigm. IEEE Comp 35(1):70–78
2. Bertozzi D, Jalabert A, Murali S et al (2005) NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. IEEE Trans Parallel Distrib Sys 16(2):113–129
3. Ge F, Wu N, Qin X, Zhang Y (2011) Clustering-based topology generation approach for application-specific network on chip. Lecture notes in engineering and computer science: proceedings of the world congress on engineering and computer science 2011, WCECS 2011, Oct 19–21, 2011, San Francisco, USA, pp 753–757
4. Ogras U, Marculescu R (2006) It's a small word after all: NoC performance optimization via long-rang link insertion. IEEE Trans Very Larg Scale Integr Sys 14(7):693–706
5. Pinto A, Carloni LP, Sangiovanni-Vincentelli AL (2003) Efficient synthesis of networks on chip. In: Proceedings of the international conference on computer design, 2003, pp 146–150
6. Srinivasan K, Chatha KS, Konjevod G (2006) Linear-programming-based techniques for synthesis of network-on-chip architectures. IEEE Trans Very Larg Scale Integr Sys 14(4):407–420
7. Srinivasan K, Chatha KS (2005) ISIS: a genetic algorithm based technique for custom on-chip interconnection network synthesis. In: Proceedings of the international conference on VLSI Design, 2005, pp 623–628
8. Leary G, Srinivasan K, Mehta K, Chatha KS (2009) Design of network on chip architectures with a genetic algorithm-based technique. IEEE Trans Very Larg Scale Integr Sys 17(5):674–687
9. Choudhary N, Gaur MS, Laxmi V, Singh V (2010) Genetic algorithm based topology generation for application specific network-on-chip. In: Proceedings of the IEEE international symposium on circuits and systems (ISCAS), 2010, pp 3156–3159
10. Liu Z, Cai J, Yao L, Du M (2009) Application-aware generation and optimization for NoC topology. In: Proceedings of the IEEE youth conference on information, computing and telecommunication, 2009, pp 259–262
11. Chang KC, Chen TF(2008) Low-power algorithm for automatic topology generation for application-specific networks on chips. IET Comp Digit Tech 2(3):239–249
12. Ar Y, Tosun S, Kaplan H (2009) TopGen: a new algorithm for automatic topology generation for network on chip architectures to reduce power consumption. In: Proceedings of the AICT, 2009, pp 1–5
13. Ge F, Wu N (2010) Genetic algorithm based mapping and routing approach for network on chip architectures. Chin J Electron 19(1):91–96
14. Yan S, Lin B(2008) Design of application-specific 3D networks-on-chip architectures. In: Proceedings of the IEEE international conference on computer design, 2008, pp 142–149