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
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 ci ∈ C represents an IP core, and each directed edge ai,j ∈ A represents the communication trace from IP ci to IP cj. Every edge has two attributes, denoted by b(ai,j) and η(ai,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 ∈ £ 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 can be mapped onto a port of a router, and the maximum number of cores mapped on a router should less than η.
2. For each ai,j ∈ A, there exists a unique path pij = (r1, r2, ..., ri) ∈ 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:

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

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).

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, \(N_{\max} = \eta - 1\). 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 \(C_1, C_2, ..., C_k\). 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 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.

---

![Diagram](image_url)

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

Fig. 1. The router structure and NoC architecture

---

![Diagram](image_url)

Fig. 2. The flow of the clustering algorithm
can be divided into four clusters C1, C2, C3, C4, as shown in Fig. 3a, and the labels of the edges in cluster C1 is mapped to port 0 in the router order according to their communication amount. Then, we three ways are attempted to recursively search the path 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) is rnext which is connected to the router r. The constructed link between router r and rnext should satisfy the constraints of the bandwidth and latency. The adjacency matrix RAdj[Mr][Mr] represents the interconnection relation among routers, where Mr 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[Mr][Mr] is checked to ensure that its value does not exceed the supported bandwidth Ω. The pot 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.

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 r1, r2, r3, r4 respectively. As shown in Fig. 4, the core c1 in the cluster C1 is mapped to port 0 in the router r1, and the cores in the cluster C2 are mapped to three ports in the router r2.

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) is rnext which is connected to the router r. The constructed link between router r and rnext should satisfy the constraints of the bandwidth and latency. The adjacency matrix RAdj[Mr][Mr] represents the interconnection relation among routers, where Mr 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[Mr][Mr] is checked to ensure that its value does not exceed the supported bandwidth Ω. The pot 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 rsrc and destination router rdest for each communication trace aij=(c1, c2).

Algorithm Output: the routing path psrc, r1, r2,..., rsrc, rdest.

Recursive terminative condition: if rnext=rsrc or find no path for the communication trace.

Recursive function: route_construction(rsrc, rdest)

1. if (rsrc==rdest) exit;
2. if (no link between rsrc and rdest)
3. if (existing empty port or port in rsrc and rdest)

   construct link between rsrc and rdest, let add aij to RAdj[rsrc][rdest], and update PortList, exit;
4. else add add aij to PathUnAssignedSet, exit;
5. else if (no empty port in rsrc)
6. rnext = get_next_rtr(rsrc);
7. if (rnext!=NULL)
8. let add aij to RAdj[rsrc][rdest];
9. route_construction(rsrc, rnext);
10. else add add aij to PathUnAssignedSet, exit;
11. else if (existing link between rsrc and rdest)
12. if (RAdj[rsrc][rdest]==∞)
13. let add aij to RAdj[rsrc][rdest], exit;
14. else
15. rnext = get_next_rtr(rsrc);
16. if (rnext!=NULL)
17. add aij to RAdj[rsrc][rdest];
18. route_construction(rsrc, rnext);
19. else add add aij to PathUnAssignedSet, exit;
20. else
21. if (existing port between rsrc and rdest)
22. if (RAdj[rsrc][rdest]==∞)
23. let add aij to RAdj[rsrc][rdest], exit;
24. else
25. rnext = get_next_rtr(rsrc);
26. if (rnext!=NULL && rnext!=rsrc)
27. let add aij to RAdj[rsrc][rdest];
28. route_construction(rsrc, rnext);
29. else if (existing empty port or port in rsrc and rdest)
30. construct link between rsrc and portnext, let add aij to RAdj[rsrc][rdest], update PortList;
31. route_construction(rsrc, rdest);
32. else add add aij to PathUnAssignedSet, exit;
33.}

Fig. 5. The pseudo code of the link construction algorithm
As an example, the number of cores in cluster C1 and C4 is the same as shown in Fig. 3b, and the communication amount of cluster C1 is 5 which are larger than that of cluster C4. As a result, the routing path for communication trace between cluster C1 and C4 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 C2 and C3, C3 and C2 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 r1 and r4, thus router r1 can be merged with router r4, 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 a 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 $\eta$ is set to be 4, and the supported bandwidth $\Omega$ 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.
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


