

Power-Aware Mapping for Reconfigurable NoC Architectures

Mehdi Modarressi, Hamid Sarbazi-Azad
Sharif University of Technology and IPM School of Computer Science, Tehran, Iran
modarressi@mehr.sharif.edu, azad@ipm.ir

Abstract

A core mapping method for reconfigurable network-on-chip (NoC) architectures is presented in this paper. In most of the existing methods, mapping is carried out based on the traffic characteristics of a single application. However, several different applications are implemented and integrated in the modern complex system-on-chips which should be considered by mapping methods. In the proposed method, the reconfiguration (which is achieved by embedding programmable switches between routers of a mesh-based NoC) allows us to dynamically change the network topology in order to adapt it with the running application and optimize the power and performance metrics. The presented network architecture can be configured as an application-specific topology, while it still holds the benefits of the regular NoC topologies such as modularity and predictable electrical properties. The experimental results show that this method can effectively adapt the NoC to the running application and improve the power consumption and performance of the system.

1. Introduction

Network-on-Chip (NoC) is a promising communication paradigm for multiprocessor system-on-chips. This communication paradigm has been inspired from the packet-based communication networks and aims at overcoming the performance and scalability problems of the shared buses in multi-core SoCs (System on Chips) [1].

Although the concepts of NoC are inspired from the traditional interconnection networks, they have some special properties which are different from the traditional networks. Compared to traditional networks, power consumption is one of the first-order constraints in NoC design [2].

The choice of network topology and mapping and routing problems are the important design issues which can dramatically affect the network performance-related characteristics such as average inter-IP

distance, total wire length, and communication flow distributions. These characteristics in turn, determine power consumption and average network latency of NoC architectures.

A key point in optimizing the NoC power/performance (which is exploited in almost all mapping methods) is to place the processing cores communicating more frequently near each other. As the number of routers between two cores reduces, the power consumption and network delay related to their communications decreases.

Almost all of the existing NoC mapping methods try to find an optimal mapping for the communication pattern of a single application [3-7]. Today, SoCs are highly complex and cost-effective chips, and as technology advances, it becomes more cost-effective to integrate several different applications onto a single SoC chip. The NoC architecture for the design should closely match the traffic characteristics and performance requirements of the different applications. As the different applications have different functionalities, the inter-IP communication characteristics can be very different across the applications. In general, a NoC that is designed to run exactly one application does not necessarily meet the design constraints of the other applications. Moreover, in programmable SoCs such as ODYSSEY [16], a synthesized NoC may be applied to run different applications which may not be known at the synthesis time and it is very likely that their traffic patterns differ from the traffics on which the mapping and synthesis is accomplished. Few NoC mapping methods have addressed mapping based on multiple applications [8, 9]. In [8], for example, a worst-case mapping method which maps the cores and optimizes the NoC based on the constraints of all the applications was proposed.

In this paper, we propose a reconfigurable architecture for regular mesh-based NoCs. It enables the network topology to dynamically match communication patterns of the currently running application. The reconfiguration is achieved by inserting several simple switches in the network which

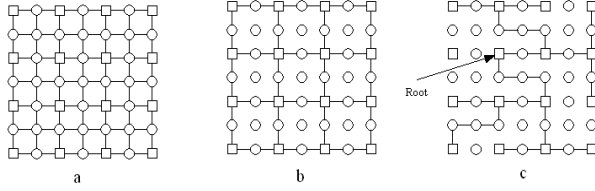


Figure 1. A reconfigurable network (a) and the network configured as (b) a mesh and (c) a binary tree

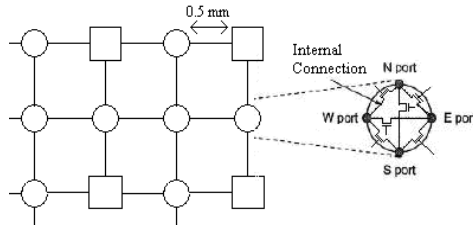


Figure 2. The internal structure of a switch

allow the network to dynamically change its connections and thus, its topology.

In this work, we first specify the set of different applications integrated onto the NoC. The physical core mapping is accomplished based on the average communication pattern of all applications. Afterwards, for each application, we use a branch-and-bound algorithm to explore a network topology which optimizes the NoC parameters of interest such as power and performance. This topology which is formed by switch configurations is loaded to the network upon starting the corresponding application.

The topologies proposed for on-chip networks vary from regular tiled-based [5], [6] to fully customized structures [7], [10], [11]. Since fully customized NoCs are designed and optimized for a specific application, they give the best performance and power results for that application. On the other hand, regular NoC architectures provide standard structured interconnects which ensures well-controlled electrical parameters. Moreover, usual physical design problems like crosstalk, timing closure, and wire routing and architectural problems such as routing and switching strategies and network protocols can be designed and optimized for a regular NoC and be reused in several SoCs. The NoC architecture proposed in this paper can be placed between these two extreme points in NoC design. While this type of NoCs can be designed and optimized like regular NoCs, they can be dynamically configured to the topology that best matches the current application.

In the following sections, we introduce the reconfigurable NoC architecture and evaluate it in

terms of power and performance gains and imposed area overheads.

2. Mapping and reconfiguration method

2.1. Dynamically reconfigurable NoCs

The NoC used in this research is a reconfigurable mesh-based interconnection network [12]. In this topology, the routers are not connected directly to each other, but rather are connected through a simple switch. These switches can be configured to establish a direct static connection between two or more incident data links. Figure 1.a shows such a network while figures 1.b and 1.c display the network configured as a mesh and a binary-tree, respectively, by properly setting the simple switches. In general, dynamic hardware reconfiguration can only be implemented on dynamically reconfigurable devices; hence, most of the reconfigurable architectures are implemented using FPGAs. Since the reconfigurable part of the proposed NoC is limited to the network switches, not only this NoC can be implemented using FPGAs, but also can be realized on ASIC platforms. The structure of the simple switch is depicted in Figure 2. The switch box consists of 6 small switches which can connect two incoming links. Tri-state gates, transmission gates, and single-transistor switches are three options for implementing such small switches. A detailed power analysis for these switching options can be found in [13]. Here, we use transmission gates as the configuration switches. Transmission gates add two NMOS and two PMOS drain capacitances to the capacitance of the link into which they are embedded. The results for these extra capacitances which are calculated using Orion power model [14] for 70 nm technology (which is the smallest feature size available in Orion) show that the capacitance of a default size transmission-gate is $6 \times 10^{-15} \text{F}$. We set the length of a wire segment to 500um (Figure 2). Therefore, switch overhead can be completely affordable due to this wire-length which is a realistic length in current NoCs.

Another consideration in the proposed topology is the long wire links which may be generated by merging a number of wire segments. This long wire may decrease the interconnection clock frequency and result in a performance overhead. To solve this problem, we can put a 1-flit buffer at the end of each wire segment in each switch box. Using such a buffer (which is inspired from pipelined circuit switching methods in conventional interconnection networks [15]) can provide pipelining over the wire and also act as a repeater for it.

We used Orion to calculate the power consumption of these 1flit-buffers for 32-bit links using 70 nm technology. Each buffer consumes a static power of 3.1×10^{-5} W. It also consumes a dynamic power of 6.4×10^{-5} W, on average. Since the network has $(2 \times (n-1))^2$ wire segments, inserting a buffer in each segment, in a 4×4 mesh for example, will result in 3.42×10^{-3} W total power which is considerably lower than the total NoC power values we present in the next section.

2.1. Mapping and reconfiguration method

In this section, we address the mapping and routing problems in the proposed reconfigurable mesh.

We first specify the set of different applications integrated onto the NoC. Any application running on a NoC is described as a *Communication Task Graph* (CTG), where each vertex represents an IP-core, and a directed edge characterizes the data transfer between the two cores. The communication volume corresponding to every edge is also provided. It is assumed that the designer has selected a set of IPs and assigned/scheduled the tasks onto these IP cores. We also assign a weight to every task-graph based on the percentage of time that the corresponding application is run on the NoC. Assigning weights enables the designer to focus more fully on major or more critical applications of the NoC. The problem is to map the cores onto different tiles of a reconfigurable mesh network and then find a topology for each task-graph and a path for every communication in the graph such that the power consumption of the NoC is minimized. The found paths should not violate the bandwidth of the network links (router input ports) to avoid congestion.

At the first step, our objective is to figure out how to map these IP cores physically onto different tiles of a mesh network such that the distance between the communicating cores is minimized. This step is accomplished for a task graph resulted from a weighted average of all the application task graphs. The vertices of the average task graph are IP cores and the edge values are calculated by taking the average of the corresponding edge values in all task graphs. If an edge does not exist in a task graph, its weight is considered to be 0 in the task graph.

The core mapping can be accomplished using an existing method based on the average graph and without taking the reconfiguration into account. Here, we have used NMAP, a power-aware core mapping algorithm presented in [6]. NMAP uses a heuristic approach that maps the task graph nodes into a mesh-based network and generates a route for every task-

Table 1. the ratio of router and link power consumptions in different loads

load	link + 1-flit buffer power	router power
0.1	0.00012	0.00058
0.2	0.00025	0.00112
0.3	0.00037	0.00173
0.4	0.00050	0.00228
0.5	0.00061	0.00291
0.6	0.00073	0.00351
0.7	0.00085	0.00409
0.8	0.00101	0.00458
0.9	0.00113	0.00512
1	0.00124	0.00572

graph edge. We use only the mapping algorithm of NMAP and then, propose a routing algorithm based on the reconfigurable network links.

Each route generated for a communication (a task-graph edge) is composed of a combination of wires and routers. In this step, we try to reduce the network hops (or number of routers) between the source and destination nodes of a high volume communication trace (or ideally connect them directly) by bypassing one or more intermediate routers by a wire. If some switch configurations (set by previous routes) do not allow the nodes to connect directly, the route can be made by a combination of routers and wires.

The dynamic power consumptions of a router and a wire of length 500um are compared in Table 1 for different traffic loads. The wire power is the sum of the power of wire, transmission-gate, and a 1-flit buffer. The router power is a calculated based on a typical router in a mesh-based interconnection network. Network load denotes the probability of receiving a flit in a cycle. Although the table compares the power of routers and links with the same load, the load of a router is the sum of the loads of its incoming links and is larger than the load of the individual links connected to it. As the table indicates, the power consumption of a wire segment is 4 to 5 times lower than a router with the same load. Consequently, from the point of view of the power consumption, it is advantageous to decrease the router loads and put the loads on the links.

In this work, route generation is done using a branch-and-bound algorithm. In this phase, the task graph edges are selected in order of their communication volumes. Selecting an unmapped edge with maximum communication volume and starting from the source router of the edge, the algorithm makes a new branch by adding a router/switch adjacent to the current node to the path. If the current node is a

router, the path can be extended by adding each of its four neighboring switches (or routers, if the router is connected to another router when the path for previous edges is established). However, the switches may have been configured by previous edges and so, can extend the path through the directions that the configuration allows. Every path has a cost which is related on the number of routers/switches the path contains. According to the aforementioned power analysis, we assign a cost of 1 to a wire segment and a cost of 5 to a router. After finding the best path for an edge, the switches of the network is set according to the found path and the algorithm continues with the next edge.

A branch (a path) is bounded (discarded) in some conditions. First, it is bounded if by adding the node, the bandwidth constraints of the newly added link is violated. In addition, if the cost of a partial path ended at a node is larger than the minimum cost of the partial paths already ended at that node or larger than the minimum cost of the completed paths, the path is discarded.

Although the branch-and-bound algorithm finds the best route for an edge, its global strategy which find route for edges in order of their communication volume is a greedy algorithm. In general, the path generated for a higher volume edge can set the switch connections in such a way that the cost of the path of a large number of other edges is increased. Therefore, this algorithm may not find the optimal configuration which minimizes the total energy consumption for a task graph. Since the problem of finding the optimal set of routings is a NP-hard [17] problem, we use a simple evolutionary algorithm to further improve the quality of solutions. In the evolutionary algorithm, the diversity in the populations is made by different routing options between the nodes. More precisely, for every edge, we select a random path among the possible shortest paths found. Each solution is represented by a set of paths generated for every edge in the task graph. Obviously, by keeping the paths we implicitly keep the configuration of the switches as well. We have a parameter for tuning the population size, p , which is 5 at present. Crossover is the only evolutionary operation we apply. This operator selects two solutions in random as parent. Afterwards, it considers the task graph edges in order of the communication volume and for every edge chooses the best route (in which the cost of communication between the sink and source nodes is minimum) between the routes in parents and copies it to the new solution.

In case of any conflict between the configuration of already configured switches in the offspring (new solution) and the currently selected best route, the

offspring gets the route from the other parent. If the problem persists, a new path for this edge is generated by the branch-and-bound algorithm explained before. A solution is evaluated based on a fitness metric which is the cumulative power generated by the traffic traversed through the network. To this end, the load of all links and routers are calculated and the power consumption of every router and link is computed according to its load using the Orion power library. The fitness is the sum of the power of all routers and links. Afterwards, if the offspring improves the power of parents, it replaces the parent with the lower quality.

Generating new solutions continues for a specified number (10 in this work) of generations. Afterwards, the best solution which consists of the switch configurations and paths is saved and will be loaded to the network upon starting the corresponding application. After loading a configuration, the switch configuration part of the solution configures the network switches and the paths in the solution set the routing table of the network routers.

3. Evaluation

To validate the performance of the proposed mapping methodology, we perform simulations on some application task-graphs as benchmark. The benchmark set includes two random graphs; random graph 1 with 25 nodes and 30 edges and random graph 2 with 25 nodes and 20 edges together with some existing SoC designs which have been used in a number of other papers: an MPEG4 decoder [10], a multi-window display (MWD) [10], an object plane decoder (OPD) [6], and 263 decoder mp3 decoder [18]. These benchmarks describe rather small-sized NoCs, but we got them from other papers on NoC mapping. Although, these SoCs have a single task graph, we generate two additional task graphs for each of them by modifying the base task graphs and integrate the three graphs into a NoC. We also assign a weight of 0.5 to the base task-graph and 0.3 and 0.2 to the other graphs. For example, the task-graph of the object plane decoder is depicted in Figure 5.a and the two additional graphs we generated are depicted in Figures 5.b and 5.c. the edge tags represent the communication volume in Mbit/s. The physical mapping is accomplished based on the average graph and then, a network topology is generated for each task-graph. Figures 5.d, 5.e, and 5.f display the topology generated for the graphs in Figure 5.a, 5.b, and 5.c, respectively.

We map the task-graphs on reconfigurable meshes with flit size of 32 bits in 70nm technology. The NoC working frequency is 250 MHz. More precisely, we set

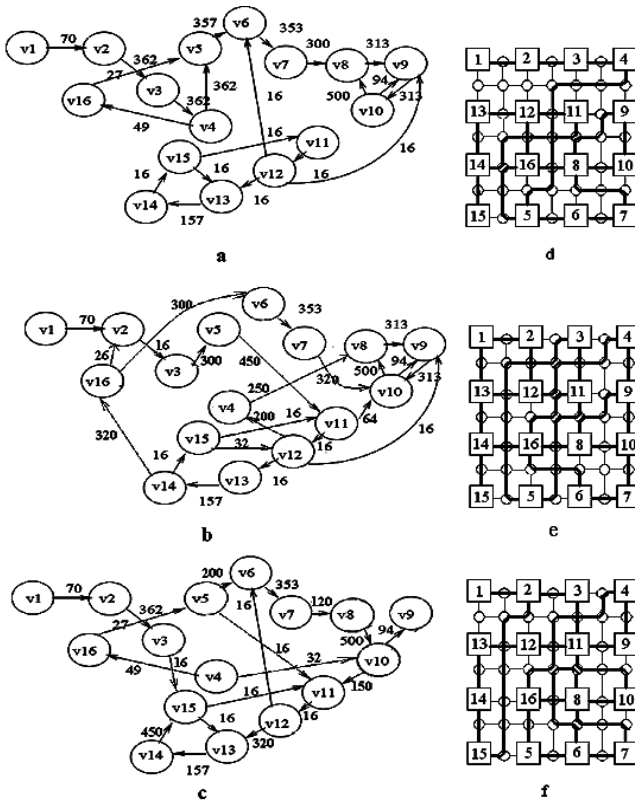


Figure 3. a) The task-graph for the object plane decoder, b) and c) two task-graphs based on the task-graph in 'a', d) the network topology for task-graph 'a', e) the network topology for task-graph 'b', f) the network topology for task-graph c.

these parameters in the Orion power library integrated in our simulator. Although the actual bit-width in NoC links may be higher than 32, we used this bit-width since most of the power components scale equally with the bit-width and so, the bit-width has no effect on comparison results. The bit-width can only affect the static power of transmission-gate switches. However, this static power changes linearly with the number of transmission-gates and by changing the number of gates from 32 to larger values (256 for example), the static power is still considerably lower than the total NoC power. As mentioned before, we estimate the power of a NoC based on the load of the routers and links of it. Table 2 displays the power consumption of the NoC for the 3 cases. The first case maps the cores based on the average task graph but does not use reconfiguration and performs the routing as in conventional fixed-connection meshes. The second and third cases use the proposed reconfigurable mapping and routing without and with evolutionary optimization, respectively (the numbers are presented in Watts). Since we intend to compare the proposed architecture with the traditional mesh we ignore some

power components that are common between two mechanisms such as router static power. However, we proposed an analysis for the static power of links in previous sub-section.

The parameters of the evolutionary algorithm are set as denoted in previous section. The results show that the reconfiguration can adapt the topology to the application and effectively reduce the power consumption of the NoC by 32%, on average. As the difference among the traffic patterns of the applications integrated into a NoC increases, the proposed reconfiguration method reduces the power consumption more effectively. For example, the two additional graphs generated for the MPEG benchmark are resulted from a slight modification to the original task-graph and so, the impact of reconfiguration on its power consumption is lower than the other benchmarks whose the additional graphs are generated with more changes. Moreover, two random task-graphs show that the impact of reconfiguration increases with an increase in the number of graph edges. When the number of edges increases, putting the source and destination of all connections near each other becomes a problem which can be alleviated by reconfiguration.

Obviously, although we focus on power consumption, reducing the distance between the cores communicating frequently can reduce the average packet latency and improve the NoC performance as well.

However, this power reduction is obtained for the price of adding extra wires. The proposed reconfigurable $n \times m$ mesh architecture has $(2m-1) \times (n-1) + (2n-1) \times (m-1)$ links (we consider two 500um wire segments as a link in order to compare them to conventional meshes) while the conventional mesh has $n \times (m-1) + m \times (n-1)$ links. To evaluate this area overhead, we estimate the area of the routers used in this research using Orion and calculate the link area based on the analysis in [19]. We scale the areas proposed in [19] to 70nm technology and for 32-bit 1mm links. The results show that the area for the 1mm-wires and the routers are 0.0204 mm² and 0.0634 mm², respectively. Based on these results, the area overhead is 19% for a 4x4 NoC and 17% for a 3x4 NoC. Like FPGAs, This overhead can be compensated by the obtained flexibility.

4. Conclusions

In this paper, we presented a core mapping mechanism for reconfigurable network-on-chip (NoC) architectures. This mechanism addresses the problem of most of the existing NoC mapping methods which

Table2. The power consumption in Watts for different mappings

Benchmarks	network size	Without reconfiguration	Reconfiguration Without GA optimization	Reconfiguration and Genetic Optimization	Ratio with and without reconfiguration
OPD	4x4	0.0457	0.0328	0.0255	0.55
MWD	4x3	0.0166	0.0096	0.009	0.54
MPEG	4x3	0.0626	0.0522	0.0521	0.83
MP3	4x4	0.1188	0.1018	0.0938	0.79
Random graph 1	5x5	0.2600	0.1790	0.1690	0.65
Random graph 2	5x5	0.1819	0.1310	0.1291	0.71

are based on the traffic characteristics of a single application. The reconfigurable NoC was formed by embedding programmable switches between routers of a mesh-based NoC.

Our evaluation results showed that compared to a conventional mesh network, this method reduces the power consumption of the NoC by 32%. Future work in this line can further consider reducing the power consumption by scaling the voltage and frequency of the routers and links based on their loads. Moreover, more optimal algorithms for mapping and routing in reconfigurable NoCs can be developed.

References

[1] L. Benini, and G. De Micheli. "Networks on Chip: A New Paradigm for Systems on Chip Design", *Design, Automation and Test in Europe (DATE)*, 2002, pp. 418–419.

[2] U. Y. Ogras, J. Hu, R. Marculescu, "Key Research Problems in NoC Design: A Holistic Perspective", *Proc. CODES+ISSS*, Jersey City, NJ, 2005.

[3] U. Y. Ogras, R. Marculescu, "Application-Specific Network-on-Chip Architecture Customization via Long-Range Link Insertion", in *IEEE/ACM Intl. Conf. on Computer Aided Design*, San Jose, 2005.

[4] J. Hu, 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, 2005.

[5] G. Ascia, V. Catania, and M. Palesi, "Multi-Objective Mapping for Mesh-based NoC Architectures," in *ISSS-CODES*, 2004, pp.182–187.

[6] S.Murali, G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," *Design Automation and Test in Europe (DATE)*, 2004, pp. 896-901.

[7] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear Programming Based Techniques for Synthesis of Network-on-Chip Architectures," in *ICCD*, 2004, pp. 422–429.

[8] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and

G. De Micheli, "Mapping and Configuration Methods for Multi-Use-Case Networks on Chips", *Asia and South Pacific Design Automation Conference - ASP-DAC*, 2006, pp. 146-151.

[9] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "A Methodology for Mapping Multiple Use Cases onto Networks on Chips," *Design Automation and Test in Europe (DATE)*, 2006, pp. 118-123.

[10] K. Srinivasan, and K. S. Chatha. "A Low Complexity Heuristic for Design of Custom Network-on-Chip Architectures", *Design Automation and Test in Europe (DATE)*, 2006.

[11] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli, "Efficient Synthesis of Networks on Chip", in *ICCD*, 2003, pp. 146–150.

[12] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill Book Co., New York, 1984.

[13] H. Wang, "A Detailed Architectural-level Power Model for Router Buffers, Crossbars and Arbiters", Technical report, Princeton University, 2004.

[14] H. Wang, X. Zhu, L. Peh and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks", *35th International Symposium on Microarchitecture (MICRO)*, Turkey, 2002.

[15] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection Networks: An Engineering Approach*, 2nd edition, Morgan Kaufmann Publishers, 2002.

[16] M. Goudarzi, S. Hessabi, A. Mycroft "No-Overhead Polymorphism in Network-on-Chip Implementation of Object-Oriented Models," *Design Automation and Test in Europe (DATE'04)*, February 2004.

[17] G.Ascia, V. Catania, and M Palesi, "An Evolutionary Approach to Network-on-Chip Mapping Problem," in *The IEEE Congress on Evolutionary Computation*, 2005, pp.112-119.

[18] K. Srinivasan, K. Chatha, and G. Konjevod "Linear Programming Based Techniques for Synthesis of Network-on-Chip Achitectures", *IEEE Tran. on VLSI*, Vol. 14, No. 4, 2006, pp. 407-420.

[19] M. Kim, D. Kim, and E. Sobelman, "NoC link analysis under power and performance constraints" *ISCAS*, Greece, 2006.