Exploiting Implementation Diversity and Partial Connection of Routers in Application-Specific Network-on-Chip Topology Synthesis

Minje Jun, *Member*, *IEEE*, Won W. Ro, *Member*, *IEEE*, and Eui-Young Chung, *Member*, *IEEE*

Abstract—This paper proposes a novel application-specific Network-on-Chip (NoC) topology synthesis method, in which the partial connection and the implementation diversity of routers are exploited. NoC has emerged as a promising solution to future system-on-chip (SoC), and many researchers have focused on the automatic synthesis of NoC topology. In our observation, those NoC topology synthesis methods resemble the logic synthesis in the following sense: both the NoC topology synthesis and the logic synthesis determine the *connections* among the components where the components are the routers in the former and the logic cells in the latter. However, an outstanding difference is that the existing NoC topology synthesis methods consider only a single implementation for each size of router, whereas modern logic synthesis tools utilize multiple implementations of a cell to produce better netlist by the feature called *technology mapping*. To tackle this drawback, we propose a novel NoC topology synthesis methodology where the implementation diversity of routers is exploited to produce optimal topologies in terms of area and/or power consumption. Two different approaches, the *post-process* approach and the *in-process* approach, are proposed for exploiting the implementation diversity to provide flexibility between synthesis time and design quality. Also, the proposed method for characterizing and modeling routers makes it feasible to consider the implementation diversity even when the partial connection of routers is considered during the synthesis. Compared to the method in which the implementation diversity is exploited but the partial connection is not, the experimental results demonstrate that the proposed method can reduce the power consumption by up to 67.8% and 40.0% on average. On the other hand, compared to the method in which the partial connection is exploited but the implementation diversity is not, the power consumption is reduced by up to 12.0% and 3.4% on average.

Index Terms—On-chip networks, system-on-chip (SoC), architecture, synthesis

1 Introduction

Due to ever-increasing complexity of System-on-Chip (SoC) integration and poor scaling of metal wire, on-chip communication architectures have been eagerly researched by both academia and industry for the past decade. Since on-chip interconnection network, also called Network-on-Chip (NoC), was proposed in [1], it has emerged as a promising solution to many-core processors and very large scaled SoCs. In fact, it was already shown that NoC is now ripe enough to be applied to real-world products [2]. The underlying philosophy of NoC is to replace the traditional ad-hoc on-chip wiring with more structured and modular architectures with *routers* and *links*. There is a variety of research topics for NoC including the topology, packet routing, router design, and application mapping, and they are well summarized and addressed in [3]. While most of the early researches on NoC had focused on general-purpose and regularly structured NoCs [1], [4], [5], recent works have shown that NoC is advantageous to the systems which are dedicated to or emphasized on specific applications, and that the customized irregular structures outperform the general-purpose regular ones for those systems [6]–[8]. In order to fully exploit the benefit of customized topology for application-specific systems, many researchers have focused on the synthesis of customized topology for target applications [8]–[11], [12]–[19]. The existing topology synthesis methods have aimed at the algorithms (or design space exploration) to find the best topology for the target application(s). They typically require several information as inputs, such as physical distances among IPs and router library which contains physical characteristics (delay, area, and power) of on-chip routers of all sizes.

In our observation, the NoC topology synthesis problem is analogous to the logic synthesis from several aspects; the logic synthesis produces *netlist* composed of various logic gates, while the NoC topology synthesis produces *network* composed of routers of various configurations. Also, modern NoC topology synthesis methods consider the physical placement of IPs (i.e. floorplan information) to accurately reflect the wire effect, which resembles the contemporary logic synthesis tools considering the cell placement to produce more realistic and better netlist (e.g. Synopsys Topographical Technology [20]). In terms of component library, pre-characterized router models are used in the NoC topology synthesis methods, which are obtained from either actual implementation [8], [13] or

- *M. Jun was with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea. He is now with System LSI Business, Samsung Electronics Co., Hwasung, Gyenggi-Do 445-701, Korea. E-mail: minje.jun@samsung.com.*
- *W. W. Ro and E.-Y. Chung are with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea. E-mail: {wro,eychung}@yonsei.ac.kr.*

Manuscript received 28 Dec. 2011; revised 07 Dec. 2012; accepted 12 Dec. 2012. Date of publication 23 Dec. 2012; date of current version 09 June 2014. Recommended for acceptance by R. Merculescu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2012.294

analytical model such as *Orion* [21], [22]. On the other hand, the logic synthesis uses the cell library provided by the foundries. However, an important point which the existing NoC topology synthesis methods are missing is that they consider only a single implementation of a router; that is, they characterized a certain configuration (e.g. the number of ports) of router with a single set of metrics such as delay, area, and power. For example, in [8], a 4×4 router is characterized to have $0.036\ \mathrm{mm}^2$ area and to consume 22.16 mW from their actual back-end implementation. However, in reality, a router can be implemented in various ways throughout the typical RTL to silicon design flow. On the contrary, in the logic synthesis, it can be easily found that a cell is designed for its various implementations, e.g. NAND2X2 to NAND2X16. Then, the synthesis tools exploit the various implementations of the cell during the procedure *technology mapping* to produce better netlist.

To tackle this point, this paper proposes a novel NoC topology design methodology which focuses on how to characterize routers and utilize the resulting router library, unlike the existing methods have concentrated mostly on the design space exploration of the topological choices. Specifically, we first propose the router characterization and modeling method, in which the building blocks of routers are separately characterized for their diverse implementations. Then, the characteristics of an entire router are obtained from those of its building blocks. Secondly, we propose the NoC topology synthesis method with two different approaches for exploiting the diverse implementations of the routers; in the post-process approach, the topology synthesis process is done first with a single implementation of the routers, and then the resulting topology is optimized just once at the end of the topology synthesis process. On the other hand, the in-process approach tries to find the best fitting implementation of every router in every candidate topology. In fact, we presented the idea of exploiting implementation diversity of routers in our previous work [23]. This work reinforces the methodology by greatly reducing the characterization efforts, which in turn makes the proposed idea (i.e. exploiting implementation diversity) applicable to the finer-grained NoC topology synthesis proposed in [24].

The rest of the paper is organized as follows. The related works are summarized in Section 2. Motivational examples and corresponding explanation are given in Section 3. The details of the proposed NoC topology synthesis methodology including router characterization and modeling and the NoC topology synthesis method are presented in Section 4. The validation of the proposed router model and the evaluation results of our NoC topology synthesis method are demonstrated in Section 5. Finally, the concluding remarks are given in Section 6.

2 Related Works

In this section, we will give a brief summary of the related works and compare them with the proposed work. Several researchers have focused on the synthesis of customized bus matrix for systems which are dedicated to or emphasized on specific applications, and they showed the advantages of the customized bus matrix over the non-customized one. S. Murali et al. proposed application-specific partial crossbar,[1] generation methods based on mixed integer linear programming (MILP) [25] and their heuristic clustering algorithm [26]. The scope of their optimization is to cluster masters and slaves to several local busses in order to minimize the central backbone crossbar. S. Pasricha et al. improved the partial crossbar synthesis method by involving more factors into their optimization scope, such as arbitration scheme and buffer size on each slave port [27], and types and sizes of system memories [28]. Although these methods greatly improved the performance and power efficiency of bus matrix architecture, they have limitation on the scalability and cannot solve the global wire problems such as long wire delay and routing congestion.

To resolve the drawbacks, many researchers attempted adoption of the switch-based network in application-specific SoCs and the custom topology synthesis methods. The topology synthesis algorithms are based on either meta-heuristic such as simulated annealing or genetic algorithm [16], [17], MILP [12], [13], and their own heuristic search algorithms [5], [8], [15], [18], [19]. Despite the differences among their algorithms, the focus of these works is to efficiently explore the search space of topological decisions of on-chip interconnection networks. In order for further improvement, the work in [24] integrated the idea of partial crossbar into the topology synthesis process and proved its effectiveness. All of these works used the physical characteristics of network components, e.g. router, obtained from either actual implementation [5], [8], [13], [15]–[17], [24] or analytical model [18] such as *Orion* [21], [22]. However, all of these works have characterized a router with a single set of metrics, and therefore missed the opportunity to exploit diverse implementations of the router.

In our previous work in [23], we have shown that exploiting diverse implementations of routers in the NoC topology synthesis can significantly improve the quality of results. However, the router characterization overhead is enlarged in the methodology since an RTL synthesis is required for every configuration of routers.[2] Even though the characterization is a one-time job and can be parallelized, the methodology in [23] is practically unapplicable to the partial connection-aware topology synthesis method in [24], especially when multi-voltage and multi-corner need to be considered.

Compared to the previous works, the contributions of our work are as follows:

- We propose a novel NoC topology synthesis methodology in which diverse implementations of routers are exploited.
- Compared to our previous work [23], we greatly reduce the router characterization effort by employing *port-level* characterization. Unlike the works in [12] and [24] which also used port-level characterization, our work considers the implementation diversity of the ports. As a result, the optimization scope of this work encompasses the advantages of our previous works, i.e. the partial connection-awareness and the implementation diversity.

1. A partially connected router (or crossbar) is a router (or crossbar) in which not all the input ports and the output ports are connected but, instead, only the necessary connections are physically established.

2. In [24], all 255 router configurations (1x2 to 16x16) were synthesized and it took about a week.

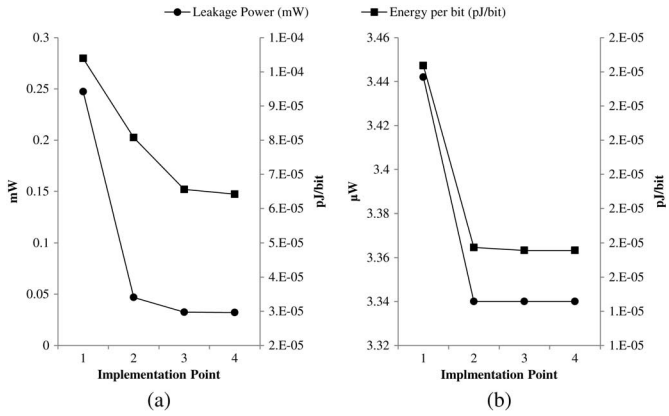

Fig. 1. Implementation diversity of (a) input port ($\texttt{fanout} = 7$) and (b) output port ($\texttt{fanin} = 3$) of an on-chip router (measured at 400 MHz operating frequency).

Note that, even though this work aims at synthesis of irregular topology, the idea can be easily adapted to any regular topology selection method such as [5]. Also, note that we do not consider floorplan and packet routing (for power reduction or deadlock avoidance) in our method since the proposed methodology can be easily integrated on the existing floorplan-NoC topology co-synthesis methods and routing allocation methods.

3 Motivation

In this section, we give examples which show the necessity of the proposed work. First, the implementation diversity of an input port and an output port is shown in Fig. 1(a) and (b), respectively. The x-axis labeled 'implementation point' indicates a set of implementation constraints; in our experiment, the longer delay constraint corresponds to the larger index of the implementation point. With the tighter (i.e. shorter) delay constraint, the resulting circuit tends to consume more leakage power and more energy for a unit bit transfer. The input port of implementation point 1 consumes more than 7.7 times leakage power and 1.6 times more energy for unit bit transfer than that of implementation point 4. Since a router consists of the ports and their connections, there exist many different ways to implement a router.

The implementation diversity of routers may affect the NoC topology as well. Fig. 2 shows one of the topology synthesis results in our experiment (specifically, G1_x4). Fig. 2(b) shows the synthesis result when only a single implementation point (implementation point 1 in the example) is used during the synthesis, where the resulting topology is a single partially connected router. Fig. 2(c) shows the result after each port in the network in Fig. 2(b) is assigned the optimal (but not in a strict manner) implementation point by our post-process optimization algorithm (see Section 4.5). In this case, the post-process optimization achieves slight power reduction from 50.43 mW to 50.10 mW without changing the topology. On the other hand, Fig. 2(d) shows the result when all the implementation points are used in the synthesis process (see Section 4.5). At the first glance, the topology is different from Fig. 2(b) and c, and the power consumption is reduced by 11.2% compared to Fig. 2(b). The reason why the solution in Fig. 2(d) cannot be found by the methods in Fig. 2(b) and (c) is that the topology in Fig. 2(d) consumes 59.53 mW (> 50.43 mW) with the implementation point 1, and therefore is discarded during the iteration of the topology synthesis. This suggests that exploiting implementation diversity of routers would enhance the quality of the NoC topology synthesis.

4 Proposed NoC Topology Design Methodology

4.1 Problem Definition and Assumptions

The problem to be tackled in this work can be classified as *application-specific irregular No Ctopology synthesis problem*. The optimization scopes of our synthesis method include 1) connections among processing elements (PEs) and routers, 2) connections among routers, 3) internal connections among the ports inside routers, and 4) implementation points of the ports inside routers.

Our synthesis process requires two inputs: 1) the building block library Lib which contains timing, area, and power information of the router building blocks (details will be discussed in Section 4.4), and 2) the *communication requirement graph (CRG)* which contains the communication requirements such as bandwidth and latency constraints of the target application, defined as follow;

- A CRG is a directed graph $G(V, E)$, where $v \in V$ denotes a source or a sink of a communication, i.e. a network interface, and $e_{i,j} \in E$ denotes a communication between $i \in V$ and $j \in V$. $bw(e)$ and $lat(e)$ are the communication volume (e.g. in MB/s) and the latency constraint (e.g. in ns) of e, respectively.

Also, we made several assumptions to simplify the problem and concentrate on the proposed ideas. The following conditions are assumed in this work:

- All the routers in the network operate at a same clock frequency/voltage, and interfacing between different clock frequency/voltage domains is done by the network interfaces.
- Each communication in the CRG is routed through only one path; that is, deterministic single-path routing is assumed for each $e \in E$.
- The communication requirements in CRG are *static*; that is, the communication characteristics described in CRG do not change over time or use-cases.[3]

With these inputs and assumptions, our NoC topology synthesis problem is defined as follow;

given CRG and the building block library Lib,

find the NoC topology (both inter- and intra-router topologies) and the implementation points for all the building blocks which yield the lowest cost (power consumption in this work),

such that all the communication requirements in CRG, i.e. $bw(e)$ and $lat(e)$ are satisfied.

4.2 Proposed Synthesis Methodology

Fig. 3 shows the proposed NoC topology synthesis methodology juxtaposed with the current logic synthesis methodology. In the logic synthesis methodology, the foundry characterizes the cells for their various implementations (i.e. various drive

3. Our method can be extended to support dynamic traffic patterns by capturing bandwidth and latency constraints for multiple time windows, similar to [15], [26].

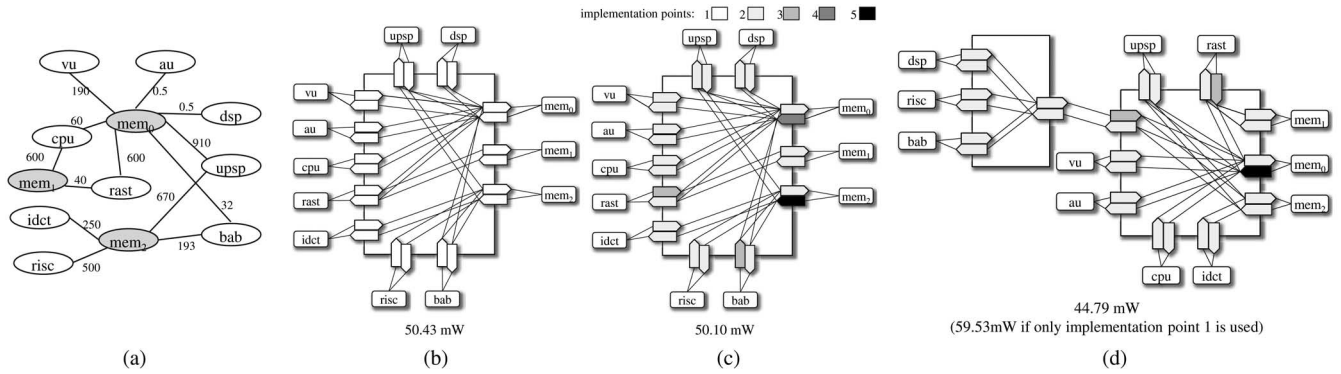

Fig. 2. Effect of considering implementation diversity in NoC topology synthesis, (a) CRG of the target system (G1_x4), (b) topology synthesized only with implementation point 1, (c) implementation points of the ports are simply replaced in the topology in (b), and (d) topology is synthesized considering all the implementation points.

strengths) and the design company uses them to produce the best netlist (shown in the left half of Fig. 3). Inspired by the logic synthesis methodology, in the proposed NoC topology synthesis methodology, the *router building blocks* (the subcomponents of on-chip router, input port and output port in this work) are characterized for their various implementations, and the resulting *building block library* is used in the topology synthesis (shown in the right half of Fig. 3).

Specifically, the router building blocks are designed by the SoC design company itself or provided by NoC IP vendors, such as ARM, Sonics, and Arteris. Provided the RTL codes of the building blocks, the design company performs the characterization of the building blocks for their various implementations with the target foundry's technology library. Together with the library of the building blocks, a *router modeling method* is needed to obtain the characteristics (delay, area, and power consumption) of a router from those of the building blocks. Since we characterize the router building blocks rather than the entire routers and obtain the routers' characteristics based on the building blocks, we can easily take the partial connection configuration of the routers into account. Finally, with the building block library and the router modeling method, the NoC topology synthesis is performed while exploiting the diverse implementations of routers to find better topological choices.

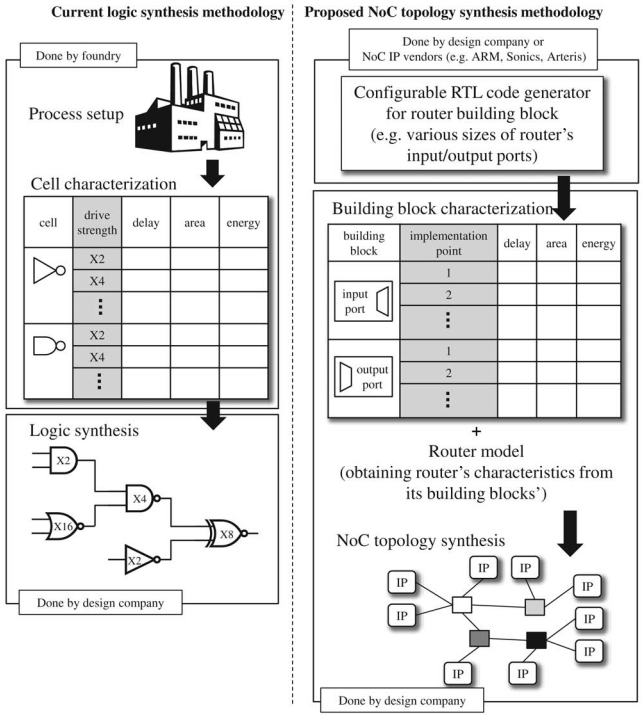

Fig. 3. The overview of the proposed NoC topology design methodology.

4.3 Router Architecture

Before we explain the proposed router characterization and modeling method, we first introduce the router architecture used in this work.[4] We designed a two-stage pipeline wormhole router, where the stages are *link traversal and input buffering (LT)* and *switch allocation and crossbar traversal (CT)*. If there is no contention, the flit which arrived the input port can be captured by the output register in the destined output port at the very next cycle. Then, the captured flit traverses the link to the downstream router after the propagation delay of the output register. We assumed source routing to simplify the router design; that is, the route is decided by the source node and just handed to the routers, and therefore the routers only need to shift the route field of the header flit when tossing it to the next router. We used stop-go flow control mechanism where the *almost full* signal of the input FIFO indicates 'stop' and 'go' of flits. The flit width is 66b where the two MSB bits indicate the types of the flit: `Head`, `Body`, `Tail`, and `Head−and−Tail` (i.e. one flit packet).

Typically, an on-chip wormhole router consists of four types of building blocks: input port, output port (or just output register), crossbar switch, and switch allocator [30] [as shown in Fig. 4(a)]. Also, a typical wormhole router is pipelined in three stages: link traversal and input buffering, switch allocation, and crossbar traversal and output buffering.[5] Note that, in

4. Note that this does not mean that the proposed methodology is restricted to the specific router architecture, but it can be generally applied to any router architectures with their proper building block classifications. For instance, the 4-stage virtual channel (VC) router in Intel's 48-core chip multiprocessor [30] can be divided into four building blocks: 1) the input buffer, 2) the block containing the switch allocator and the route computation unit (the two operations are done in the same pipeline stage), 3) the block containing VC allocator and the DEMUX part of the crossbar switch (VC allocation and buffer read are done in the same pipeline stage), and 4) the MUX part of the crossbar switch and the output register.

5. Separate routing stage is typically omitted when source routing or look-ahead routing is used [31], [32].

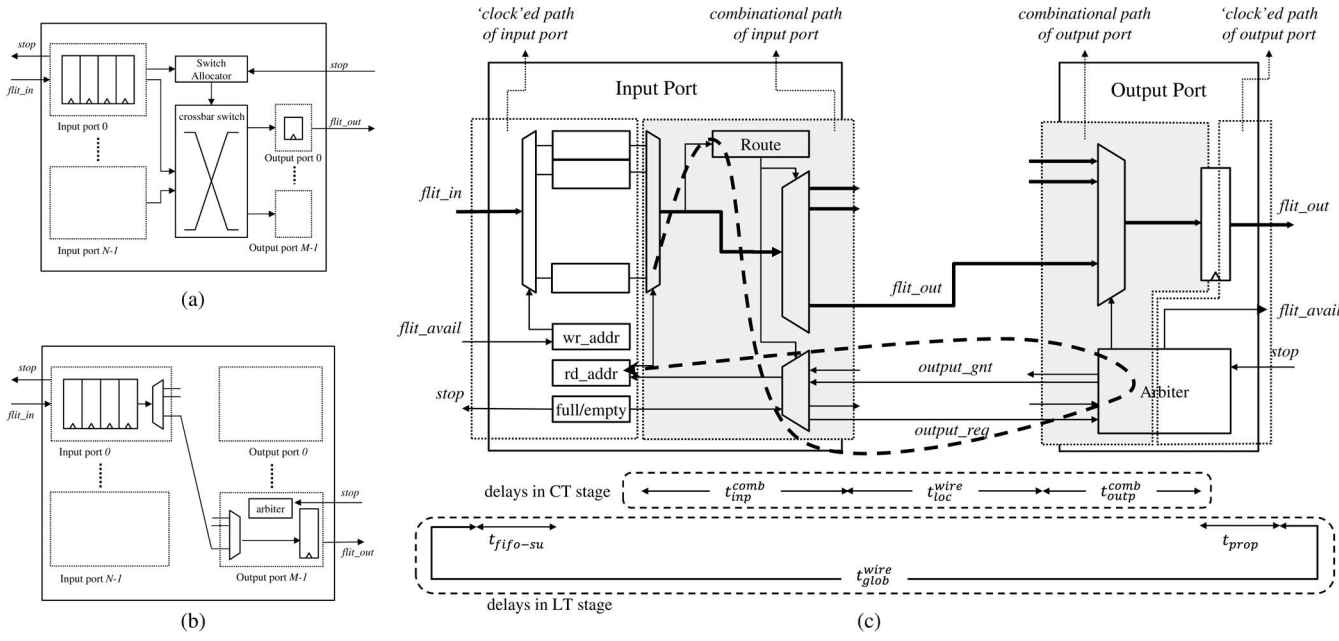

Fig. 4. The architectures of router and building blocks: (a) a typical wormhole router architecture, (b) the proposed building block classification and router architecture, and (c) the detailed architectures of the input port and the output port.

our router design, it is not only meaningless but also increases the characterization efforts to characterize all these four types of building blocks separately, since these building blocks are not dedicated to their own pipeline stages (see Section 4.4). In order to simplify the characterization process, we reduced the types of building blocks from four to two, *input port* and *output port*, by 1) using distributed arbiters over all output ports rather than a centralized switch allocator, and 2) splitting the crossbar switch into demultiplexor (DEMUX) part and multiplexor (MUX) part and putting them into input ports and output ports, respectively. The modified router architecture and the details of the input port and the output port are shown in Fig. 4(b) and (c), respectively. Only local wire bundles reside among the ports to connect them.

The input port consists of flit FIFO and its control logics (which is also in charge of the stop-go flow control and the output port request), a route computation block, and MUX and DEMUX data paths. The route computation block is in charge of updating the selection signals of the DEMUX and MUX only when the current flit is a `Head` flit, and in charge of shifting the route field of the flit. The output port consists of an arbiter, MUX data path, and an output register. A flit is forwarded according to the following procedure:

- At the beginning of the LT stage, a flit captured in an output port of an upstream router begins to traverse the link to the downstream router, after the propagation delay of the output register (t_{prop}).
- The flit traverses the long global wires between the upstream and the downstream routers (t_{glob}^{wire}).
- The flit which arrived the downstream router is successfully stored in the input buffer after the setup-time of the FIFO ($t_{fifo-su}$), and this is the end of the LT stage.
- The flit captured in the input buffer becomes available for the route computation, if it is at the head of the FIFO, after the propagation delay of input FIFO. This is the beginning of the CT stage. If the flit is a `Head` flit, the routing information is updated and otherwise kept as the previous value. With the routing information, the flit and the request signal are sent to the target output port. The delay of this path is t_{inp}^{comb}.
- In the output port, the arbitration is done for the received request signals and the grant signals are sent back to the input ports to update the FIFO's read pointers. At the same time, the flit passes through the MUX according to the updated arbitration result, and reaches the output register. This is the end of the CT stage, and the delay of this path is t_{outp}^{comb}.

Note that the paths in the CT stage are mostly implemented in combinational logic with some exceptions; updating the read pointers in the input FIFO and updating the token in the arbiter are in the clocked path. The critical path of the CT stage is indicated with a bold dashed line in Fig. 4(c). In the following subsection, we will introduce the router building block characterization method for the presented router architecture.

4.4 Building Block Characterization and Router Modeling

Based on the architectures of the router and the building blocks presented in the previous subsection, we perform characterization only for the input ports and the output ports of their various sizes. All the possible router configurations, including partial connections, can be covered by simple assembly of the ports. The size of an input port is measured by its *fanout*, the number of the output ports to which it is connected. Similarly, the size of an output port is measured by its *fanin*, the number of the input ports to which it is connected.

As shown in Fig. 4(c), each type of port has *clock path* and *combinational path* inside. The combinational path of the input port and that of the output port together comprise a clock path. However, since we characterize them separately, each half of the CT stage should be characterized as combinational path in its belonging port.

The characterization flow takes the following parameters as inputs:

- Set of available clock frequencies Φ, where $\phi \in \Phi$ denotes a possible clock frequency.
- Set of fractional values Ψ, where $\psi \in \Psi$ indicates the delay of the combinational path of the port as the fraction of the clock period.
- The maximum values of the fanout and the fanin, denoted as FO and FI, respectively.

In our previous work [23], the entire router is characterized only for the clock path and it is assumed that any arbitrary clock frequency is possible in the network. However, in real designs, possible clock frequencies are limited due to the implementation overheads, and therefore we assume that the possible clock frequencies are given by the designer as an input. The fractional value $\psi \in \Psi$ determines the delay of the combinational path of the port as the ratio to the given clock period $1/\phi \in \Phi$. If the combinational path delay of an input port t_{inp}^{comb} is $0.3 \times t_{clk}$ (t_{clk} is the clock period), then that of an output port t_{outp}^{comb}, which is connected to the input port, must be less than or equal to $0.7 \times t_{clk}$ so that the total delay of the CT stage is within a clock cycle. We define the pair (ϕ, ψ) as *Implementation Point* of the port, and denote it with $\upsilon \in \Upsilon$. In the rest of the paper, the implementation point will be denoted as the pair (ϕ, ψ) or υ interchangeably.

The overall characterization flow is shown in Fig. 5. Each type of port is characterized for all possible sizes (i.e. fanout or fanin) and for all the possible combination of clock frequencies and combinational path delays. The RTL generator is invoked for each type and each size of the port, and RTL synthesis is performed multiple times for every combination of $\phi \in \Phi$ and $\psi \in \Psi$. After each synthesis, the result is checked for the constraint violation and, if there is no violation, saved into the building block library. The total number of RTL synthesis needed is simply $FO \times FI \times |\Phi| \times |\Psi|$. For example, if the maximum allowable size of the router is 10x10 and there are 5 possible clock frequencies and 4 combinational path delays, we need 400 RTL synthesis (200 for each type of port). Note that all the possible partial connection configurations of routers can be covered with this characterization. Compared to the previous work, the characterization method used in [23] requires 499 RTL synthesis for 1x2 to 10x10 routers (omitting 1x1 router) only for fully connected routers, and the number dramatically increases when the partial connection is considered. In addition to the reduction in the number of RTL synthesis, each synthesis time is greatly shortened since the component being synthesized is a port which is much smaller than an entire router in the proposed method.

At each implementation point, a port is characterized for its area and power consumption as long as the delay constraints are met. The power consumption of a port at implementation point υ, P_υ is modeled as follow.

$$P_\upsilon = leak_\upsilon + \alpha_\upsilon \times f_{clk} + \beta_\upsilon \times \tau \times f_{clk}, \quad (1)$$

where f_{clk} and τ are the clock frequency (measured in MHz) and the activity (measured in MB/s) of the port, respectively. The first term $leak_\upsilon$ is the leakage power consumption which is independent of the clock frequency and the activity. The second term is one part of dynamic power consumption which is proportional only to the clock frequency (e.g. clock network), and α_υ is the corresponding coefficient. The last term is the other part of dynamic power consumption which is proportional to both the clock frequency and the input activity (e.g. register) and β_υ is the corresponding coefficient. The three power-related coefficients, $leak_\upsilon$, α_υ, and β_υ are extracted throughout the characterization process and saved into the building block library.

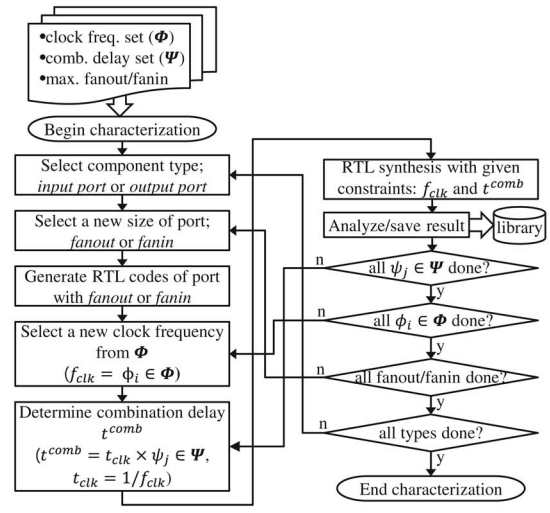

Fig. 5. The building blocks characterization flow.

With the delay, area, and power consumption obtained for the ports, the area and power consumption of a router can be easily calculated by summing those of the ports inside the router. Formally, A_r^{router} and P_r^{router}, the area and power consumption of a router r, respectively, are

$$A_r^{router} = \sum_{p \in P_r} A_p^{port},$$
$$P_r^{router} = \sum_{p \in P_r} P_p^{port}, \quad (2)$$

where $p \in P_r$ is a port in the router r, and A_p^{port} and P_p^{port} are their area and power consumption, respectively.

As aforementioned, the path delay of the CT stage is modeled as the sum of t_{inp}^{comb} and t_{outp}^{comb}. This router modeling may show inaccuracy due to several factors; for example, the effect of the local wires between ports is not accounted, and the exact values of the output loads (driving strengths) of an input port (output port) are hard to obtain since the port is synthesized without considering how the boundary of the output port (input port) is synthesized. However, in our experiment, the proposed router modeling shows only about 3% error compared to the measured values for the entire router. The details of the validation methodology and results will be demonstrated in Section 5.2.

4.5 Synthesis Algorithm

4.5.1 Top Level Synthesis Flow

We embedded the proposed ideas in the $CEP + ERGF$ network topology synthesis framework presented in [15]. For the sake of self-containedness, we include brief introduction to the CEP + ERGF; it is an irregular on-chip network topology synthesis algorithm for specific applications. In the CEP + ERGF, the design space is defined by the *chained edge partitioning (CEP)*, in which a network topology is represented

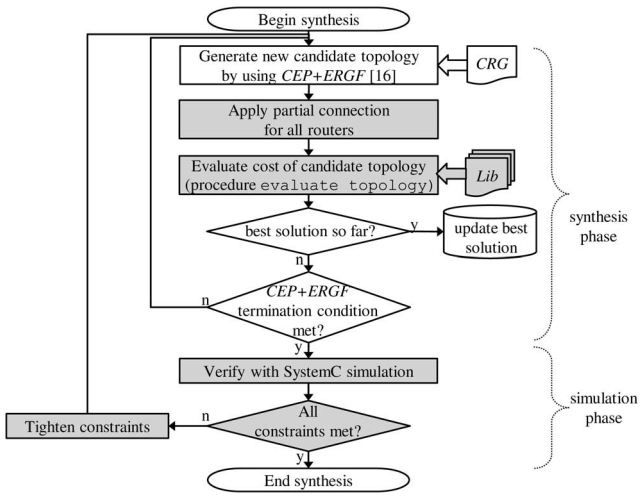

Fig. 6. The top level topology synthesis flow.

by a set of edge partitions. A topology is encoded as a set of non-negative integer sequences produced by the *enhanced restricted growth function (ERGF)*. The algorithm iteratively generates candidate topologies by the ERGF and evaluates them until the ERGF sequence reaches its maximum. Since the work in [15] considers only the inter-router level topology and single implementation point, we added the features of applying partial connection and exploiting implementation diversity into the CEP + ERGF.[6]

Finally, the top-level synthesis flow is shown in Fig. 6. In Fig. 6, the steps in shaded boxes are added or modified parts from the work in [15]. There are four major changes made in this work: 1) the step for removing unused connections in routers is added, 2) the evaluation step is enhanced so that every building block in the candidate topology is evaluated with its optimal implementation point assigned, 3) packet latency model is enhanced to reflect contention latency, and 4) simulation-based verification phase is added.

Since removing unused connections in the routers is quite straightforward, we will skip the detailed explanation for it. For the added verification phase, an in-house cycle-count-accurate SystemC simulator is used in which PEs are mimicked by traffic generators. If any bandwidth or latency violation is detected during the simulation phase, the synthesis phase is invoked again with more tightened constraints; specifically, $bw(e)$'s in CRG are multiplied or $lat(e)$'s are divided by a *constraint multiplier* to make the synthesis phase to estimate the bandwidth and/or latency more conservatively.

In the rest of this subsection, we will first present the details of the modified evaluation algorithm which also performs the implementation point assignment (Section 4.5.2). After that, we will discuss the modified latency model (Section 4.5.3).

4.5.2 Implementation Point Assignment and Topology Evaluation

We propose two different approaches for exploiting the implementation diversity in the NoC topology synthesis:

6. More specifically, `CEP_rand` algorithm in [16] is used as inter-router level topology synthesis algorithm. Note that the proposed idea can be easily augmented with other iterative NoC topology synthesis algorithms, such as the algorithms in [17]–[20].

Procedure: evaluate_topology
Input: Topology T, Port library Lib
Output: $Cost_T$

```
1:  φ = get_required_frequency(T);
2:  Cost_T = 0;
3:  foreach r ∈ R
4:    foreach p ∈ P_r
5:      p →set_implementation_point(φ, ψ_min);
6:      evaluate_port(p, Lib);
7:    endfor
8:    sort_by_cost(P_r);
9:    Cost_r = 0;
10:   foreach p ∈ P_r
11:     p →squeeze_down_comb_path(φ, Lib);
12:     Cost_r += evaluate_port(p, Lib);
13:   endfor
14:   Cost_T += Cost_r;
15: endfor
```

Fig. 7. The procedure to evaluate a topology.

Procedure: squeeze_down_comb_path
Input: ϕ, Lib

```
1:  ψ_ub = find_max_allowable_ψ(P_r^connected);
2:  Cost_best = INFINITY;
3:  foreach ψ ∈ Ψ (ψ ≤ ψ_ub)
4:    Cost = calculate_cost(φ, ψ, Lib);
5:    if (Cost < Cost_best) then
6:      Cost_best = Cost;
7:      ψ_best = ψ;
8:    endif
9:  endfor
10: this→ ψ_fixed = true;
```

Fig. 8. The procedure to assign the best implementation point to the combinational path of a port.

the *post-process* approach and the *in-process* approach. In the former, the topology synthesis process is done first with a single implementation for the routers, and then the resulting topology is optimized just once at the end by substituting the ports with their best-fitting implementations. Therefore, using diverse implementations does not affect the determination of the topology at all. On the other hand, the latter approach tries to find the best fitting implementation of every port for every candidate topology. Since all the candidate topologies are evaluated with their optimal implementations assigned, using the diverse implementations can affect the determination of the topology with the in-process approach. In the rest of this subsection, the in-process approach will be introduced in detail, and then the post-process approach can be understood straightforwardly.

The detailed algorithm for `evaluate_topology` at line 6 of Fig. 6 is shown in Fig. 7. The procedure `evaluate_topology` starts by obtaining the minimum required clock frequency ϕ from the given topology T (line 1). Since we assume the single clock frequency for the network, this step determines ϕ for all

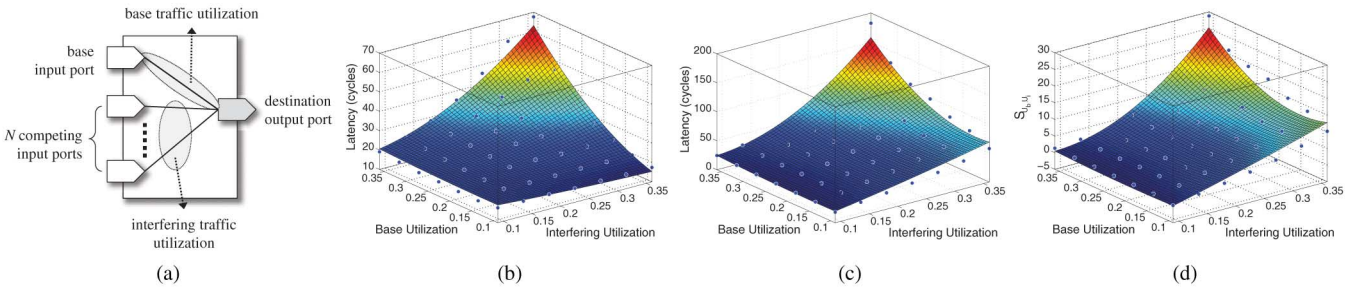

Fig. 9. Latency modeling, (a) modeling environment, (b) regressed latency when $N_c = 1$, (c) regressed latency when $N_c = 5$, and (d) regressed S_{U_b,U_i}.

the ports in the current network candidate. The following procedure is for determining ψ for each port. The ports cannot be optimized independently since there exist connections between the combinational paths of the input ports and those of the output ports, and for all the connected paths, $t_{inp}^{comb} + t_{outp}^{comb}$ must be less than or equal to the clock period. Therefore, finding the optimal assignment of ψ for all the ports is not a simple problem. In this work, we use a *greedy* approach in assigning ψ for each port. First, from line 4 to 7, the minimum value of ψ, ψ_{min} is assigned for all the ports and they are evaluated. In other words, all the ports are evaluated at their fastest implementations for the combinational paths. In the next step, at line 8, the ports are sorted in a *descending* order according to their evaluated power consumption. At line 11, the procedure `squeeze_down_comb_path` is performed to assign ψ for each port according to the sorted order. This procedure replaces the combinational path of the port with its least power-consuming possible implementation (figuratively, squeezes down the port to its smallest possible state). The rationale behind this approach is to replace the most power-consuming port with its least power-consuming implementation. More detailed explanation will be given later in this subsection. Now that the port is assigned the proper ψ, its implementation point is fixed and therefore it is evaluated again for the new implementation point (line 12).

The details of `squeeze_dow_comb_path` at line 11 of Fig. 7 is shown in Fig. 8. At the beginning of the procedure, it first determines how much the port can be squeezed down (line 1); the port may be connected to one or more ports of the opposite type which are already assigned their ψ. In this case, the port may not be assigned its least power-consuming implementation since the combinational path delay must be small enough to satisfy the clock frequency when combined with the combinational paths of the connected ports ($P_r^{connected}$). Therefore, the upper bound of ψ for this port, ψ_{ub} is obtained first by searching the ports to which it is connected. Within the upper bound, all possible ψ is tested to find the least-power consuming implementation from line 3 to 9. Note that we test all the possible ψ rather than taking the ψ_{ub} for the port for the sake of better generality of the algorithm; a larger ψ means longer delay of the combinational path of the port, and an implementation with longer (i.e. loose) delay constraint tends to consume less power. However, this tendency cannot be guaranteed especially when the implementation relies on the automated design flows (e.g. RTL synthesis and auto P&R). After the best ψ for this port, ψ_{best} is found, the procedure ends by setting the flag "ψ_fixed" to `true`. This flag is used when another port connected to this port tries to determine its ψ_{ub} (line 1). More specifically, a port with "ψ_fixed = true" returns its assigned ψ when called by the procedure `find_max_allowable_`ψ, whereas a port with "ψ_fixed = false" returns ψ_{min} so that the requesting port can be freely squeezed down without upper bound.

In the post-process approach, the procedure `evaluate_topology` is performed with only a single implementation for each port during the topology synthesis, while skipping the procedure `squeeze_down_comb_path`. Then, at the end of the procedure `NoC_topology_synthesis`, the procedure `evaluate_topology` is invoked again with using, at this time, all the implementations and the procedure `squeeze_down_comb_path`.

The additional complexity of the proposed method can be broken down into the following three components: 1) pre-evaluation of the ports (line 6 in Fig. 7), 2) sorting the ports (line 8 in Fig. 7), and 3) squeezing down the combinational paths of the ports (line 11 in Fig. 7). To analyze the complexity of the three components, let the maximum number of ports on a router be N_P. Then, the complexity of evaluating all the ports is $O(N_P)$ since evaluation of one port can be done with $O(1)$ complexity. The complexity of sorting the ports is typically $O(N_P log N_P)$. The complexity of the third component is the multiplication N_P and the complexity of the procedure `squeeze_down_comb_path`. The complexity of `squeeze_down_comb_path` is $O(N_P)$ (line 1 in Fig. 8) $+O(|\Psi|)$ (line 3 to 9 in Fig. 8). Accordingly, the additional complexity of the proposed method is $O(N_P) + O(N_P log N_P) + O(N_P \times (N_P + |\Psi|)) = O(max(N_P^2, N_P \times |\Psi|))$. Note that these additional computations occur only once at the end of the synthesis phase in the post-process approach, whereas they occur in every candidate topology evaluation step in the in-process approach.

4.5.3 Modified Latency Model

The work in [15] used the simple hop-count level latency estimation which cannot capture the contention latency. That simple latency model may produce obsolete results especially for the application having high communication volume. Therefore, we changed the latency model to reflect the contention latency.

We follow the latency modeling methodology used in [32] with some modifications. Same as in [32], we model the packet latency as a function of *base traffic utilization* and *interfering traffic utilization*; suppose a situation where an input port P_i in a router competes for an output port P_o with the other input ports of the router, as shown in Fig. 9(a). Also, suppose that we are interested in the latency that P_i experiences to P_o. In such a situation, we call P_i as *base input port* and its utilization of P_o as

TABLE 1
Regression Results

	c_0	c_1	c_2	c_3	c_4
$Lat^1_{U_b,U_i}$	21.85	-7.75	-27.12	24.35	3020
S_{U_b,U_i}	-5.64	23.75	69.29	-349.9	2545

base traffic utilization (denoted as U_b). The other input ports of the router that compete for P_o with P_i are named as *competing input ports* and their total utilization of P_o is named as interfering traffic utilization (denoted as U_i).[7]

We made two major changes to the latency model used in [32]: 1) we reflect the dependence on the number of competing input ports (denoted as N_c hereafter) and 2) use higher order equation for the regression. We calculate the latency with a specific N_c ($Lat^{N_c}_{U_b,U_i}$) with the following regression model.

$$Lat^{N_c}_{U_b,U_i} = c_0 + c_1U_b + c_2U_i + c_3U_bU_i + c_4U_b^2U_i^2, \qquad (3)$$

where c_0 to c_4 are the fitting coefficients.[8] The regression results when $N_c = 1$ and $N_c = 5$ are shown in Fig. 9(b) and (c), respectively. The results show the obvious dependence of the latency on the number of competing input ports. We observed that the latency increases linearly as N_c increases at specific (U_b, U_i). We define the slope at (U_b, U_i) as S_{U_b,U_i}. We also observed that the dependence of S_{U_b,U_i} on U_b and U_i is similar to Lat_{U_b,U_i}. Therefore, we use the same regression model in Eqn. 3 for S_{U_b,U_i} and the result is shown in Fig. 9(d).

Finally, we estimate the packet latency with the following equation.

$$Lat_{N_c,U_b,U_i} = Lat^1_{U_b,U_i} + (N_c - 1) \times S_{U_b,U_i}. \qquad (4)$$

The obtained fitting coefficients for $Lat^1_{U_b,U_i}$ and S_{U_i,U_b} are listed in Table 1.

5 Experiment

5.1 Experiment Environment

The evaluation results of the proposed method are presented in this section. In the first part of the experiment, we will validate the proposed router characterization and modeling method. In the second part, we will demonstrate the effectiveness of the proposed NoC topology synthesis method which exploits the partial connection and the implementation diversity of routers. The router characterization process is fully automated by our in-house Verilog RTL and characterization script (for synthesis, and timing and power analysis) generators. The *topographical synthesis* is performed for the RTL synthesis to minimize inaccuracy of the frontend-only characterization, using Synopsys Design Compiler and 90 nm design kit. Detailed synthesis environment is summarized at the top of Table 2. The timing and power analysis is done using Synopsys PrimeTime. The parameters for the router architecture and characterization are shown at the middle and bottom of Table 2, respectively. The entire characterization process, from the RTL and script generation to building the complete port library, was done within roughly a day using single core virtual machine with 512 MB main memory. The host machine has 2.8 GHz processor and 8 GB main memory.

TABLE 2
Configurations of Router and Its Characterization

RTL synthesis environment	
Tech. library	Synopsys 90nm
P/V/T points	typical/typical (1.2V)/all (multi-V_{TH})
Other options	- register-level clock gating - topographical synthesis - flatten design
Router characterization	
Φ (unit: Mhz)	{400, 600, 800, 1000}
Ψ	{0.2, 0.4, 0.6, 0.8}
FO/FI	10/10
Router configuration	
flit width	66bit
input FIFO depth	8 flits
'almost_full' threshold	2

TABLE 3
Benchmark Description

Symbol	$\|V\|$	$\|E\|$	Description
G1	12	13	mpeg4 decoder [5]
G2	16	21	multimedia SoC [14]
G3	17	20	mobile multimedia player [18]
G4	19	22	mobile application processor [18]
G5	46	88	game SoC [18]

For the evaluation of our NoC topology method, we applied the method to the five benchmarks used in [5], [13], [17]. We multiplied the communication volumes of the benchmarks with three different factors so that the wider range of the implementation points can be exploited during the synthesis.[9] The characteristics of the benchmarks are summarized in Table 3 where the columns labeled $|V|$ and $|E|$ show the numbers of PEs and communication edges, respectively. In the simulation phase of our method, we assume that packet size is 512b (flit size is 64b, same as the data width of a port). Therefore, a packet consists of 1 head flit and 8 payload flits.

5.2 Validation of Router Modeling

In order to validate the proposed port-based router modeling method, we compare the values from our router modeling to those from the actual measurement. The methodology used to validate the proposed modeling is shown in Fig. 10(a). Using the actual implementations of the ports, a top-level router RTL code can be simply *assembled* by selecting the ports of the desired sizes and implementation points, and connecting each other. In the next step, we set the ports as "don't touch" modules since they are already in their synthesized form (i.e. netlist) and we want their implementations not to be changed. After that, we perform synthesis for the router, which will likely end up with optimizing only the local wires between the ports. Finally, the characteristics of the resulting router implementation are measured and compared to the values from the model.

The router configuration and the implementation points of its ports are shown in Fig. 10(b), and the clock frequency and

7. Refer to [33] for the formal definitions of the base traffic utilization and the interfering traffic utilization.

8. The latency model in [33] does not have the last term of the righthand-side of Eqn. 3.

9. The original benchmarks have not enough amount of traffic to exploit the clock frequencies in Φ other than the minimum clock frequency (400 MHz).

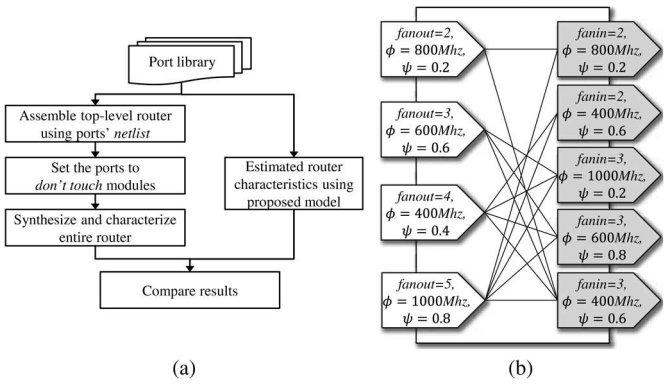

Fig. 10. Router model validation, (a) the validation method and (b) the router configuration and implementation points of the ports.

the input bandwidth (which determines toggle rate) assumed are 400 MHz and 600 MB/s, respectively. The results are shown in Table 4. The third row, labeled as "Delay", is for the longest CT stage delay [the path indicated by the bold dashed line in Fig. 4(c)] among the CT stages paths between the input ports and the output ports. The dashed rectangle in Fig. 10(b) indicates the path which has, by measurement, the longest CT stage delay. Note that the expected longest CT stage delay is 2.5 ns between the 3rd input port and the 5th output port ($2.5 \times 0.4 + 2.5 \times 0.6$), but the actual path and the delay value is different from the expectation. It is because the the expected delay is given as the *upper bound* for the RTL synthesis and therefore the resulting implementation can have any smaller delay than expected, in case of successful synthesis. The result shows that the accuracy of the proposed model is roughly about 97% for all the metrics (specifically, from 96.813% to 97.621% in absolute values). We believe that the result convinces the validity of the proposed router modeling method including the building block characterization presented in Section 4.4.

5.3 Evaluation of the Topology Synthesis Method

The effectiveness of our NoC topology synthesis method presented in Section 4.5 is demonstrated in this subsection.[10] We compared the *post-process* approach and the *in-process* approach to the conventional approach which considers only a single implementation for the routers. Specifically, the following six cases will be compared:

- The in-process optimization is used but the partial connection of routers is not exploited (**NOPAR + INP**).
- The partial connection is exploited but only a *single* implementation point with $\phi = 1000\ MHz$ is used (**SIG1000**).
- The partial connection is exploited but only a *single* implementation point with $\phi = 800\ MHz$ is used (**SIG800**).

10. Due to the space limitation, we present only the most important subset of the results in the paper. To compensate for this limitation, a supplement document (which can be found in the Computer Society Digital Library at https://doi.ieeecomputersociety.org/10.1109/TC.2013.294) is prepared which contains more details of the reported results as well as sets of other experimental results. In our experiments, no case was found where the bandwidth/latency constraint violation is detected in the simulation phase. The measured latencies can be also found in the supplement document. The location of the document is http://dtl.yonsei.ac.kr/others/jun_exploiting_supplement.pdf.

TABLE 4
Router Model Validation Results

Metric	Measured	Model	Error (%)
Area (mm^2)	0.136	0.141	3.187
Delay (ns)	1.739	1.730	0.493
Power (mW)	25.157	24.573	-2.379

- The post-process optimization is performed after SIG1000 (**SIG1000 + PO**).
- The post-process optimization is performed after SIG800 (**SIG800 + PO**).
- The partial connection is exploited and the in-process optimization is used (**INP**).

Note that SIG1000 and SIG800 represent for the method in [24] in which the partial connection of switches is exploited in the synthesis process but the implementation diversity is not considered. On the other hand, NOPAR + INP represents for the method in [23] in which the implementation diversity is exploited but the partial connection of routers cannot be considered due to the limitation of its router characterization and modeling method. For SIG1000 and SIG800, $\psi = 0.4$ and $\psi = 0.6$ are assumed for the input ports and the output ports, respectively.

We first compare NOPAR + INP and INP to show the effectiveness of exploiting the partial connection of routers together with the implementation diversity, and this comparison directly shows the advantage of our method over the method in [23]. The power consumptions of NOPAR + INP and INP are shown in Fig. 11, where values are normalized to those of NOPAR + INP. The numbers (with the prefix 'x') shown above the benchmark symbols are the bandwidth multiplication factors used to artificially multiply the communication volumes of the benchmarks. The results show that the power consumption is reduced by up to 67.8% (G5_x2) and 40.0% on average. This reduction in power consumption can be achieved by 1) reducing the number of routers used and 2) minimizing the sizes of ports inside the routers.

Secondly, we discuss the effectiveness of exploiting the implementation diversity by comparing SIG1000, SIG800, SIG1000 + PO, SIG800 + PO, and INP. Note that the partial connection is considered in the same way for all the five approaches. The total power consumption of the five approaches is shown in Fig. 12, where the values are normalized to those of SIG1000. The results show that applying only the post-process optimization brings considerable amount of power reduction compared to the single implementation approaches. Quantitatively, the post-process optimization

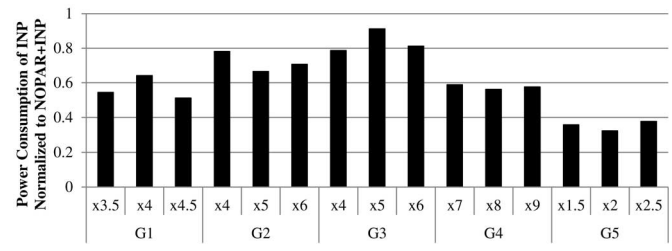

Fig. 11. Effect of considering the partial connection together with the implementation diversity.

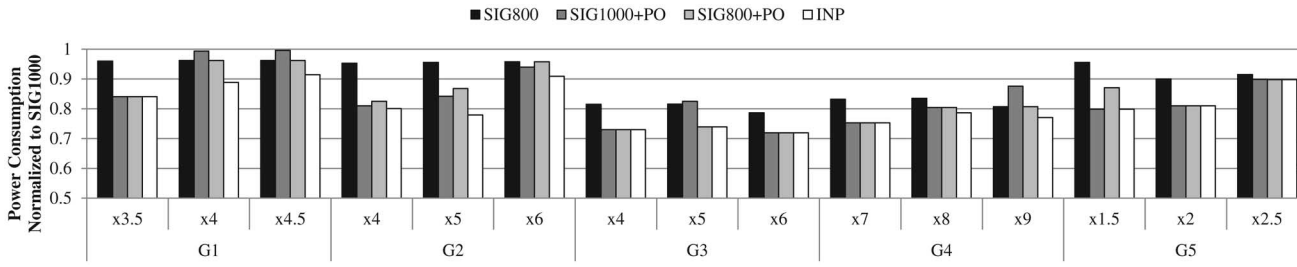

Fig. 12. Effect of exploiting the implementation diversity and evaluation of the two different approaches: the post-process and the in-process. The values are normalized to those of SIG1000.

reduces the power consumption by up to 28.1% (G3_x6) and 15.8% on average when applied to SIG1000. When applied to SIG800, the reduction is up to 13.4% (G2_x4) and 6.5% on average. Note that the power reduction by the post-process optimization is achieved solely by replacing the implementation points of the ports, while the network topology and the internal connections of the routers are not affected.

Using the in-process optimization, the power reduction is up to 28.1% (G3_x6) and 19.1% on average compared to SIG1000, and up to 18.5% (G2_x5) and 9.4% on average compared to SIG800. Also, even compared to the post-process optimization, the in-process optimization shows further power reduction. The power reduction is up to 12.0% (G4_x9) and 10.1% (G2_x5), and 3.7% and 3.1% on average, compared to $\mathrm{SIG1000} + \mathrm{PO}$ and $\mathrm{SIG800} + \mathrm{PO}$, respectively. The in-process optimization often produces different topologies compared to the post-process optimization. More specifically, the in-process optimization shows improvement over the post-process optimization by choosing different topologies for G1_x4, G1_x4.5, G2_x5, G2_x6, G4_x8, and G4_x9. This result proves that our idea, exploiting implementation diversity, does help improve the quality of NoC topology synthesis process.

The proposed method may increase the synthesis time since it requires the additional implementation point assignment procedure. In our experiment, we found that the additional time for the post-process optimization is unmeasurably small, but the in-process method increased the synthesis time by up to 249% (G1_x4.5) and 57.8% on average. Therefore, the post-process optimization can be preferably used if the synthesis time is critical, and the in-process optimization can be used if the quality is the most crucial factor.

6 Conclusion

In this paper, we proposed a NoC topology synthesis methodology which exploits diverse implementations of on-chip routers. Thanks to the proposed port-based router characterization and modeling method, the partial connection of routers can be effectively considered during the topology synthesis, and at the same time, each individual port can take advantage of exploiting its multiple implementations. The conventional router characterization methods cannot support either the implementation diversity nor the partial connection in the topology synthesis, whereas the proposed router characterization and modeling method covers both properties. The validation results show that our router model is accurate enough (within 3% error rate). In the evaluation of our NoC topology synthesis method, it is shown that the proposed method can considerably improve the design quality; compared to the method in which the implementation diversity is exploited but the partial connection is not, the proposed method can reduce the power consumption by up to 67.8%; compared to the method in which the partial connection is exploited but the implementation diversity is not, the amount of power reduction is by up to 28.1%.

We decided not to take the floorplain effect into account in this work, and left it as our future work; we plan to extend the proposed work to a co-synthesis method of the NoC topology and the floorplan where the implementation diversity of *links* can be considered together. Also, we plan to validate the method in more dynamic traffic environment.

Acknowledgment

This work was supported in part by Basic Science Research Program through the National Research Foundation (NRF) funded by the Ministry of Education (2013R1A1A2011208), by the IDEC and by Samsung Electronics.

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. Des. Autom. Conf.*, 2001, pp. 684–689.
[2] G. De Micheli, C. Seiculescu, S. Murali, L. Benini, F. Angiolini, and A. Pullini, "Networks on chips: From research to products," in *Proc. Des. Autom. Conf.*, 2010, pp. 300–305.
[3] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NOC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 1, pp. 3–21, Jan. 2009.
[4] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Des. Test Comput.*, vol. 22, no. 5, pp. 414–421, Sep./Oct. 2005.
[5] S. Murali and G. De Micheli, "Sunmap: A tool for automatic topology selection and generation for NoCs," in *Proc. Des. Autom. Conf.*, 2004, pp. 914–919.
[6] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits Syst. Mag.*, vol. 4, no. 2, pp. 18–31, 2004.
[7] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, no. 2–3, pp. 105–128, 2004.
[8] S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. De Micheli, and L. Raffo, "Designing application-specific networks on chips with floorplan information," in *Proc. Int. Conf. Comput.-Aided Des.*, 2006, pp. 355–362.
[9] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli, "Efficient synthesis of networks on chip," in *Proc. 21st Int. Conf. Comput. Des.*, 2003, pp. 146–150.
[10] U. Y. Ogras and R. Marculescu, "Energy-and performance-driven noc communication architecture synthesis using a decomposition approach," in *Proc. Conf. Des. Autom. Test Eur.*, 2005, pp. 352–357.

[11] U. Y. Ogras and R. Marculescu, ""It's a small world after all": NoC performance optimization via long-range link insertion," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 7, pp. 693–706, July 2006.
[12] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear-programming-based techniques for synthesis of network-on-chip architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 4, pp. 407–420, Apr. 2006.
[13] M. Jun, S. Yoo, and E. Y. Chung, "Mixed integer linear programming-based optimal topology synthesis of cascaded crossbar switches," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2008, pp. 583–588.
[14] M. Jun, S. Yoo, and E. Y. Chung. "Topology synthesis of cascaded crossbar switches," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 6, pp. 926–930, June 2009.
[15] M. Jun and E. Y. Chung, "Design of on-chip crossbar network topology using chained edge partitioning," *Comput. J.*, vol. 53, no. 7, p. 904, 2010.
[16] J. Yoo, D. Lee, S. Yoo, and K. Choi, "Communication architecture synthesis of cascaded bus matrix," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2007, pp. 171–177.
[17] J. Yoo, S. Yoo, and K. Choi, "Topology/floorplan/pipeline co-design of cascaded crossbar bus," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 8, pp. 1034–1047, Aug. 2009.
[18] S. Yan and B. Lin, "Application-specific network-on-chip architecture synthesis based on set partitions and steiner trees," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2008, pp. 277–282.
[19] J. Chan and S. Parameswaran, "Nocout: NoC topology generation with mixed packet-switched and point-to-point networks," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2008, pp. 265–270.
[20] Synopsys. *Topographical technology* [online]. Available: http://www.synopsys.com/tools/implementation/rtlsynthesis/dcultra/Pages/default.aspx
[21] H. S. Wang, X. Zhu, L. S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *Proc. 35th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2002, pp. 294–305.
[22] A. B. Kahng, B. Li, L. S. Peh, and K. Samadi, "Orion 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *Proc. Conf. Des. Autom. Test Eur.*, 2009, pp. 423–428.
[23] M. Jun, S. Yoon, and E. Y. Chung, "Exploiting multiple switch libraries in topology synthesis of on-chip interconnection network," in *Proc. Conf. Des. Autom. Test Eur.*, 2010, pp. 1390–1395.
[24] M. Jun, D. Woo, and E. Y. Chung, "Partial connection-aware topology synthesis for on-chip cascaded crossbar network," *IEEE Trans. Comput.*, vol. 61, no. 1, pp. 73–86, Oct. 2010.
[25] M. Srinivasan and G. De Micheli, "An application-specific design methodology for stbus crossbar generation," in *Proc. Conf. Des. Autom. Test Eur.*, 2005, pp. 1176–1181.
[26] S. Murali, L. Benini, and G. De Micheli, "An application-specific design methodology for on-chip crossbar generation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 7, pp. 1283–1296, July 2007.
[27] S. Pasricha, N. Dutt, and M. Ben-Romdhane, "Constraint-driven bus matrix synthesis for MPSoC," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2006, pp. 30–35.
[28] S. Pasricha and N. Dutt, "Cosmeca: Application specific co-synthesis of memory and communication architectures for MPSoC," in *Proc. Conf. Des. Autom. Test Eur.*, 2006, pp. 700–705.
[29] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl et al., "A 48-core ia-32 message-passing processor with dvfs in 45 nm cmos," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC'10)*, 2010, pp. 108–109.
[30] A. Banerjee, P. T. Wolkotte, R. D. Mullins, S. W. Moore, and G. J. M. Smit, "An energy and performance exploration of network-on-chip architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 319–329, Mar. 2009.
[31] J. Howard, S. Dighe, S. R. Vangal, G. Ruhl, N. Borkar, S. Jain et al., "A 48-core ia-32 processor in 45 nm CMOS using on-die message-passing and dvfs for performance and power scaling," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, Jan. 2011.
[32] Y. Jang, J. Kim, and C. M. Kyung, "Topology synthesis for low power cascaded crossbar switches," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 12, pp. 2041–2045, Dec. 2010.

Minje Jun (M'08) received the BS, MS, and PhD degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2006, 2008, and 2013, respectively. He is now with System LSI Business of Samsung Electronics. His research interests include system-on-chip architecture and network-on-chip with the special emphasis on their design automation.

Won Woo Ro (M'05) received the BS degree in electrical engineering from Yonsei University, Seoul, Korea, in 1996, and the MS and PhD degrees in electrical engineering from the University of Southern California, Los Angeles, in 1999 and 2004, respectively. He worked as a research scientist in the Electrical Engineering and Computer Science Department, University of California, Irvine. He currently works as an associate professor in the School of Electrical and Electronic Engineering, Yonsei University. Prior to joining Yonsei University, he has worked as an assistant professor in the Department of Electrical and Computer Engineering, California State University, Northridge. His industry experience also includes a college internship at Apple Computer, Inc., and a contract software engineer in ARM, Inc. His research interests inlude high-performance microprocessor design, compiler optimization, and embedded system designs.

Eui-Young Chung (SM'99–M'06) received the BS and MS degrees in electronics and computer engineering from Korea University, Seoul, Korea, in 1988 and 1990, respectively, and the PhD degree in electrical engineering from Stanford University, California, in 2002. From 1990 to 2005, he was a principal engineer with SoC R&D Center, Samsung Electronics, Yongin, Korea. He is currently a professor with the School of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea. His research interests include system architecture and VLSI design, including all aspects of computer-aided design with the special emphasis on low-power applications and flash memory applications.