On-Chip Decentralized Routers with Balanced Pipelines for Avoiding Interconnect Bottleneck

Ryota Yasudo
Keio University
Yokohama, Japan 223–8552
yasudo@am.ics.keio.ac.jp

Hiroki Matsutani
Keio University
Yokohama, Japan 223–8552
matutani@arc.ics.keio.ac.jp

Michihiro Koibuchi
National Institute of Informatics
Tokyo, Japan 101–8430
koibuchi@nii.ac.jp

Hideharu Amano
Keio University
Yokohama, Japan 223–8552
hunga@am.ics.keio.ac.jp

Tadao Nakamura
Keio University
Yokohama, Japan 223–8552
nakamura@pipelining.jp

ABSTRACT

Technology scaling makes designers face difficulties dealing with wire delay of long global interconnects, especially for high-radix networks. In this context, we propose decentralization of on-chip packet routers. A decentralized router consists of submodules, each of which has particular functionality and they are scattered on a link, thereby long wires are segmented. Our starting point is from a conventional router architecture, and we illustrate four case studies to generalize our proposal. We also propose a new buffer design and how to balance pipelines of a router. A proof-of-concept is shown in 28-nm process technology. Our results demonstrate that the decentralization of an on-chip router enables Link Traversal (LT) stages to be eliminated, and the critical path delay is improved by up to 45% with the reduced area compared with a conventional router. As technology advances, the benefit of the decentralized routers become more substantial in the nano-scale era.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Multiple data stream architectures—*Interconnection architectures*

General Terms

Design, Performance

Keywords

Decentralized router, delay model, interconnect bottleneck, interconnection networks, router architecture, wire delay

1. INTRODUCTION

It is 14 years since Ho *et al.* forecasted that designers would face difficulties dealing with wire delay [6]. Unlike gate delay, wire delay of long global interconnects increases with process scaling owing to wire resistance and capacitance. This forecast has already become real, and researchers call it the "interconnect bottleneck". For instance, a recent study [2] on Network on Chip (NoC) reports that technology scaling creates the interconnect bottleneck in both delay and energy. Although NoC provides high communication bandwidth and parallelism as an efficient solution for many-core chips, wire delay limits its performance even when we use recent process technologies. A conventional way to reduce wire delay is inserting many repeaters, but there is still a limit to this and besides it consumes an unacceptably large amount of energy [6]. In this work, we radically take a different way to address the interconnect bottleneck.

NOCS '15, September 28–30, 2015, Vancouver, BC, Canada
Copyright 2015 ACM. ISBN 973-1-4503-3396-2/15/09 ...$15.00
DOI: http://dx.doi.org/10.1145/2786572.2786583

Wire delay in NoC leads us to a new clash between logical and physical distances, that is, the number of hops and delay from a source to a destination. On this occasion, we investigate low-latency topologies for high-radix networks that include long-range links, such as Flattened Butterfly [8]. They reduce the number of hops to a destination effectively, whereas wire delay per hop increases on account of long-range links. Consequently, their well-balanced implementation becomes increasingly difficult as technology evolves. Another case of our interests is small-world NoCs [10]. They add random links to a standard mesh topology to induce the small-world phenomenon, thereby reducing average latency as well as improving achievable throughput. There is an ultimate trade-off between large wire delays and these topological performance gain, as the device scaling continues to grow. Through our observation, we should carefully consider both logical and physical distances when we discuss network latency on future and recent NoCs.

This paper introduces decentralization of on-chip routers, which is a universal methodology to solve aforementioned problems caused by wire delay. A decentralized router is composed of plural submodules that each of have a particular functionality and they are scattered on a link, thereby long wires would shorten. In addition, we propose a novel buffer design on decentralized architecture. We implement the design in 28-nm process technology and evaluate gate delay and wire delay to demonstrate the efficiency of the decentralized router architecture. In this regard, we formulate a delay model, optimize arrangement of submodules, and evaluate the critical path delay.

Section 2 surveys prior research related to decentralized router architecture. Section 3 introduces architectures of the conventional and decentralized routers, and formulates delay models. In Section 4, we show the simulation results, and then balanced pipelines optimize decentralized routers. Finally, Section 5 concludes the paper.

2. RELATED WORK

Several decentralized router architectures have been proposed for different purposes. To the best of our knowledge, Rotary Router (RR) [1] is the first to show signs of decentralized architecture for on-chip networks. RR provides no crossbar switch or arbiter. Instead, it has distributed modules on two independent rings, which force packets to circulate either clockwise or anti-clockwise, traveling from port to port. It eliminates head-of-line blocking to improve the performance. Its architecture and concept are very distant from ours, but this study reveals the potential of decentralized architecture.

Afterward, moving to the nano-scale era, distributed switch architecture [11] presents the idea of reducing the negative impact of links by decentralized architecture, which is similar to our concept. It improves the trade-off between the power consumption and the operating frequency, that is, it increases the maximum operating frequency with the reduced peak power consumption. The characteristic of this study is that its starting point is from a modular switch, which is a specific architecture. On the other hand, our starting point is from canonical router architecture because we aim at generalization of decentralized router architectures.

The latest implementation of decentralized architecture is ElastiNoC [12]. This study introduces decentralization of routers based on Virtual Channel (VC), which allow for traffic separation and isolation to enable deadlock avoidance and improve network performance. Moreover, it provides a scalable distributed self-testing mechanism. This mechanism enables testing sessions to be conducted in a modular manner over multiple phases and achieves high fault coverage. Applying decentralized architecture to making fault-tolerant networks is an original idea here.

These preceding studies make us appreciate that decentralized architecture is beneficial, although there are differences in architecture and concept. We generalize decentralized architectures using four case studies of common routers as well as follow the precedents, especially the concept of reducing the impact of wire delay. Furthermore, we propose an alternative approach: decentralized buffer design and optimization of the arrangement of modules based on balanced pipelines.

3. ARCHITECTURE AND DELAY MODELS

3.1 Baseline router

First, we describe a baseline conventional router. Figure 1 shows an overview of our baseline virtual-channel router. It consists of n input channels, an arbiter, a crossbar switch, and m output channels. If the topology is 2D-mesh, both n and m are five for connecting to neighboring routers and a local core. An input channel provides multiple VCs, each of which has an input buffer and Route Computation (RC) logic that computes the next route by using routing algorithm such as dimension order routing or west-first routing. For input buffers, FF-based FIFO buffers or RAM are used. Since we target high performance routers, we adopt FF-based FIFO buffers and virtual cut-through flow control. An arbiter allocates a pair of the output channel and the VC for each incoming packet on the basis of the state of the next router that is sent through output channels. A crossbar switch consists of m n-to-1 multiplexers, each of

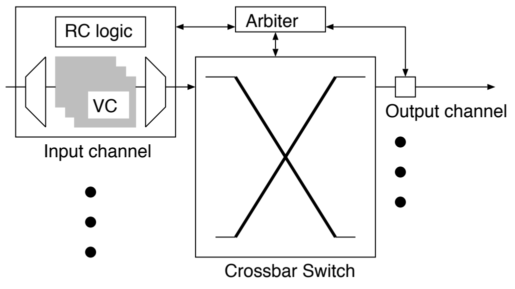


Figure 1: Overview of baseline router architecture.

which is controlled by a select signal from the arbiter. In an output channel, there is a register that can store one flit.

The routing processing typically consists of four steps: Route Computation (RC), Virtual channel Allocation (VA), Switch Allocation (SA), and Switch Traversal (ST). The baseline router is a conventional three-cycle router. That is, it has three pipeline stages: RC, VSA, and ST. In the VSA stage, VA and SA are speculatively performed in parallel. If the VA fails, SA will be ignored even though it succeeds. In addition, Link Traversal (LT) is also required to transfer a flit to the next router, and consequently four cycles are required in total. Since the latency of a NoC is directly related to the pipeline depth in a router, LT stages limit the performance. To make matters worse, the latency of LT stages increases as technology progress because it is determined by wire delay. For this reason, LT stages are obvious drawbacks to the conventional router.

The delay model of the baseline router is formulated as follows:

$$T_{\mathrm{RC}} = \max(G_{\mathrm{fifo}_{\mathrm{wr}}}, G_{\mathrm{rc}}), \tag{1}$$

$$T_{\mathrm{VSA}} = G_{\mathrm{arb}}, \tag{2}$$

$$T_{\mathrm{ST}} = G_{\mathrm{fifo}_{\mathrm{rd}}} + G_{\mathrm{cb}}, \tag{3}$$

$$T_{\mathrm{LT}} = W_{\mathrm{link}}, \tag{4}$$

where T, G, and W are the delay of each pipeline stage, the gate delay of each processing, and the wire delay of the link between routers, respectively. The critical path delay of the router T_{router} is determined by the maximum T as follows:

$$T_{\mathrm{router}} = \max(T_{\mathrm{RC}}, T_{\mathrm{VSA}}, T_{\mathrm{ST}}, T_{\mathrm{LT}}). \tag{5}$$

Naturally, the best performance is achieved by balancing pipeline stages. We will show the measured value and balance pipeline stages in Section 4.

3.2 Naive decentralized router

We decentralize the baseline router to reduce the negative impact of LT stages. We divide the router function into small ones and design submodules for each function. Since router pipelining processes each step in a clock cycle at a dedicated stage, each function corresponds to each pipeline stage. The basic idea is to segment the function of a router into several modules in this manner and intersperse them with a link. Then a link between routers is divided into shorter links, and LT stages are segmented. Segmented wire delay is distributed to the remaining pipeline stages: RC, VSA, and ST. Figure 2 shows the naively decentralized router. The decentralized router consists of three submodules, the functions of which are explained below.

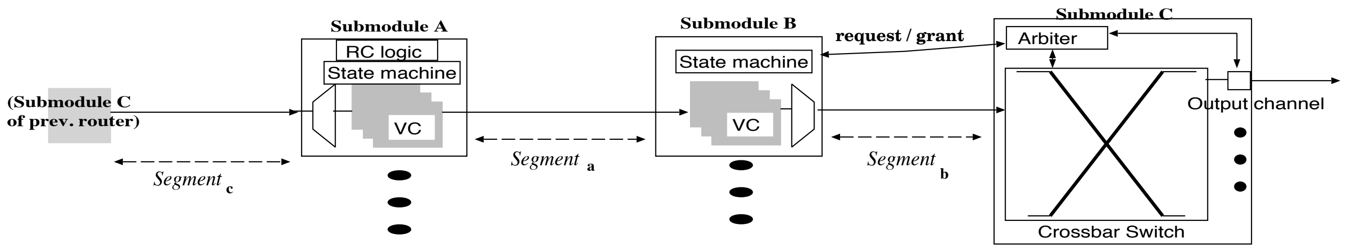


Figure 2: Overview of naive decentralized router architecture.

Submodule-A. The function of the submodule-A is to compute routes and generate requests to an arbiter. Initially, a state machine waits for a header flit. For a header flit, RC logic finds a route and sends a request signal, called an initial request, to an arbiter. At the same time, submodule-A wakes up submodule-B by sending an enable signal. Then the state is changed to transfer state, and the state machine waits for a tail flit. After it arrives, the state is initialized.

Submodule-B. This is the simplest submodule: its function is to forward packets when a grant is received. It has no particular hardware. Initially, a state machine waits for the enable signal sent by submodule-A. If the enable signal arrives, the state is changed to the transfer state, and the state machine waits for a tail flit in the same way as submodule-A. When a pipeline does not stall, a grant against the initial request immediately comes. When a pipeline stalls, however, the submodule-B generates a request different from the initial request.

Submodule-C. This is the biggest submodule: it consists of a crossbar switch, an arbiter, and output channels. It is arranged at the same position of the conventional router, and the other submodules are dispersedly arranged on a link. Output channels monitor the state of virtual channels of the next router and send this information to an arbiter. On the basis of this, the arbiter arbitrates between input channels and asserts a grant signal. Then multiplexers of the crossbar switch are configured to let packets pass through.

At this point, we find important problems associated with decentralization. Buffers are required in each submodule, and hence there is highly frequent buffering, which obviously increases power consumption. Specifically, the dynamic power consumption of the buffers increases $(n-1)$ times (n represents the number of submodules). In our case, it doubles because n is three and buffering occurs in RC and VSA stages. Since the dynamic power in a router increases rapidly as the network traffic increases, this is a matter of consequence. Buffers also increase latency in a pipeline stage at read or write operations. Meanwhile, area does not change because buffers are only separated. We must reduce the buffer cost for designing the router suitable for NoC. This is not only an inherent problem in a decentralized router but also general one. For example, implementing long-rage links requires inserting buffers in the repeaters [10].

Note that the flow control also changes in the case of the naive architecture. This is because each submodule has buffers and flow control is performed individually between submodule-A and submodule-B. Since each buffer is half the size of the baseline router, the naive architecture degrades performance at heavy network load, and saturation throughput lowers. This problem is resolved by using a decentralized buffer design, which is shown below.

The delay model of the naive decentralized router is formulated as follows:

$$T_{\mathrm{RC}}=\max(G_{\mathrm{fifo_{wr}}}, G_{\mathrm{rc}}) + W_{\mathrm{segment_a}}, \tag{6}$$

$$T_{\mathrm{VSA}}=\max(G_{\mathrm{fifo_{wr}}}, G_{\mathrm{arb}} + 2W_{\mathrm{segment_b}}), \tag{7}$$

$$T_{\mathrm{ST}}=W_{\mathrm{segment_b}} + G_{\mathrm{fifo_{rd}}} + G_{\mathrm{cb}} + W_{\mathrm{segment_c}}, \tag{8}$$

where W_{segment} is the wire delay of each segmented link as shown in Figure 2, and consequently $W_{\mathrm{segment_a}}+W_{\mathrm{segment_b}}+W_{\mathrm{segment_c}} = W_{\mathrm{link}}$. Since segmented wire delays are determined by the arrangement of submodules, we can optimize it on the basis of the delay model. We discuss it in Section 4.

This delay model reveals another problem, which is caused by request and grant in VSA stages. Arbitration is required to forward request and send back a grant serially within one clock cycle, and consequently larger wire delay $(2W_{\mathrm{segment_b}})$ is needed (*cf.* Equation 2). Since T_{VSA} is typically large as evaluated in Section 4, the VSA stage becomes an apparent bottleneck, which is inadmissible. We therefore propose the method to overcome drawbacks of the naive decentralized router in the next subsection.

3.3 Novel decentralized router using new buffers

We propose separating a control path from a data path. Packets go through links that consist of only buffers, and control information that includes flit type, request, and grant goes through links that consist of the decentralized router submodules. This method avoids increasing the amount of buffering and removes buffer delay completely from pipeline stages. For this data path approach, we implement a new buffer design that reduces buffer costs. Figure 3 illustrates the new buffer design. It consists of D latches and is arranged dispersedly on a link. A bunch of D latches whose size corresponds to one flit composes a column and flits transit columns. Since columns are decentrally organized, wire delay arises between adjacent columns. If we design the buffer so that the delay between the first and last columns becomes two cycles, data can be transferred within the same cycles as the buffering duration of the conventional three-cycle router. This design is, so to speak, the delay line consisting of D latches.

Certainly D latches achieve better delay, area, and power consumption than those of flip-flops, but they are difficult to control. Thus, the control of D latches is essential. For instance, recent studies propose novel buffer design using D latches such as Elastic Buffer [9] and Marching Memory Through-type [14]. In the present study, an additional mechanism for accepting pipeline stalls is needed for the buffer, otherwise packet loss occurs when packets conflict. We therefore add back pressure as an external signal. It is trans-

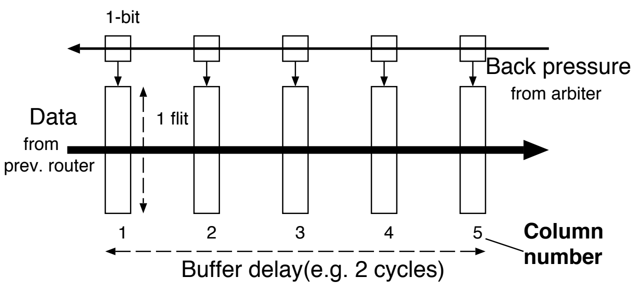


Figure 3: Decentralized buffer composed of D latches.

ferred in a direction opposite to data at the same speed as data. It passes D latches that are added to each column and stops data transfer as shown in Figure 3. When an arbiter wants to stop data (i.e., packets conflict), arbiter asserts back pressure. Then back pressure comes through in order, from the header flit.

The design is implemented in Verilog-HDL, and then synthesized, placed and routed by Synopsys Design Compiler and Synopsys IC Compiler. Data are transferred with two clock cycles when back pressure is low and are stored in D latches after the arbiter asserts back pressure. After back pressure is negated, data transfer restarts. This method spontaneously transfers data correctly without performance overhead.

Based on the above, the proposed decentralized router architecture is illustrated in Figure 4. It is basically identical to the naive design, but buffers are disjoined as shown in this figure. The columns of the buffer are placed at regular intervals. Flits from the previous router are transferred to the data path, and control signals are picked up and proceeded to the control path. Then data transfer and control processing are handled independently.

In contrast to the naive architecture, a flow control does not change in the case of the proposed architecture, because the buffers are separated from all the submodules. The identical flow control that the baseline router uses, namely virtual cut-through, is performed. Consequently, performance of the proposed and the baseline architectures is the same on a cycle level. This means the performance is determined by the critical path delay rather than execution cycles.

The delay model of the proposed router is formulated as follows:

$$T_{\mathrm{RC}} = G_{\mathrm{rc}} + W_{\mathrm{segment_a}}, \tag{9}$$

$$T_{\mathrm{VSA}} = G_{\mathrm{arb}} + W_{\mathrm{segment_b}}, \tag{10}$$

$$T_{\mathrm{ST}} = G_{\mathrm{cb}} + W_{\mathrm{segment_c}}. \tag{11}$$

T_{VSA} is improved compared with the naive architecture. This is because the grant delay is eliminated, or rather, back pressure replaces grant. Grant needs the wire delay $W_{\mathrm{segment_b}}$, whereas back pressure does not, because the arbiter is adjacent to the head column of the buffer. Furthermore, $W_{\mathrm{segment_b}}$ becomes also unnecessary in T_{ST}, because data arrive at the submodule-C before ST stages.

Besides the delay of each pipeline stage, the delay of data T_{data} must be considered in the case of the proposed router. It is formulated as follows:

$$T_{\mathrm{data}} \approx G_{\mathrm{buffer}} + \frac{W_{\mathrm{link}}}{C}. \tag{12}$$

Table 1: Routers used in four case studies.

#	Routing algorithm	# of VCs	Pipeline structure
1	DOR	2	[RC][VSA][ST][LT]
2	DOR	8	[RC][VSA][ST][LT]
3	West-first	2	[RC][RS][VSA][ST][LT]
4	Duato's protocol	2	[RC][RS][VSA][ST][LT]

G_{buffer} is the critical path delay of the buffer, and C is the number of cycles that correspond to a duration of buffering. The critical path delay of the proposed router becomes the maximum T including T_{data}.

3.4 Case study

We now get back to the case study since our study targets general rather than specific router architecture. We select four common router architectures, and present how to decentralize them. After that, we elucidate the effect of decentralization in each case and discuss the difference between these cases and the preferred router architecture. Table 1 outlines the routers used in the four case studies.

3.4.1 The simple router

The simple router completely corresponds to the baseline router whose routing algorithm is Dimension Order Routing (DOR). We assume a very simple router that has only two VCs per input channel that are allocated with fixed priority. Specifically, the next VC is computed by RC logic and packets never change VCs to pass. We also use this router for area evaluation.

3.4.2 The router with many VCs

The second case adds many VCs to the simple router. Since many VCs improve efficiency of resource allocation by allowing more packets/flits to participate in arbitration [15], this case is common. Router architecture and routing algorithm are essentially the same as before, but eight VCs are allocated with round-robin priority in this case. Thus, the critical path of VSA becomes long. Many VCs are implemented only in this case.

3.4.3 The partially adaptive router

So far we have assumed deterministic routing, but from here we adopt adaptive routing. In this case, router architecture changes, for adaptive routing requires additional hardware, such as selection function. It is inserted between an input channel and an arbiter as shown in Figure 5. It receives multiple requests from input channels and selects one of them. The selected request is sent to an arbiter. In this case, an additional pipeline stage called route selection (RS) is implemented. Its delay is formulated as follows:

$$T_{\mathrm{RS}} = G_{\mathrm{rs}} + W_{\mathrm{segment_d}}. \tag{13}$$

$W_{\mathrm{segment_d}}$ is the segmented wire delay between submudule-B and submodule-D (*vid.* Figure 5b).

We employ minimal west-first routing as partially adaptive routing. It is based on the turn model [4] to make it deadlock free. This algorithm first routes a packet west, and then adaptively in other directions. We assume a round-robin mechanism as its selection algorithm: that is, the selection function selects an output channel in rotation, rather than considering network congestion. Consequently, the router architecture becomes simple for an adaptive router.

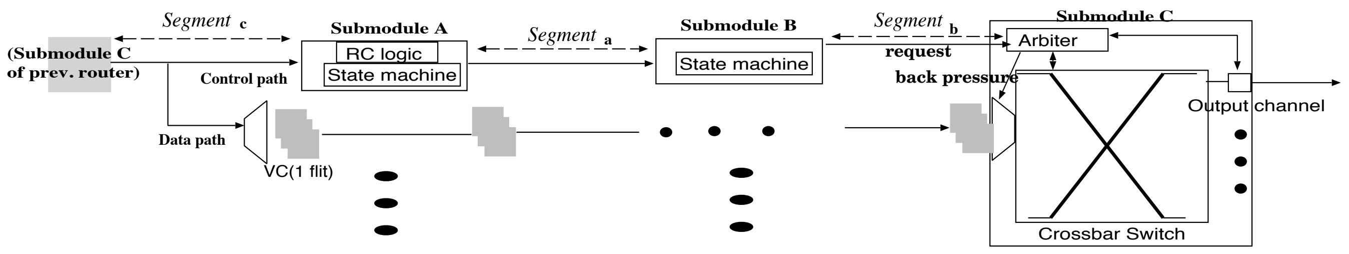
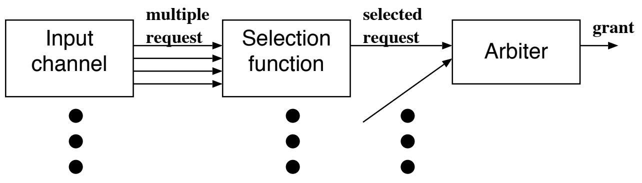
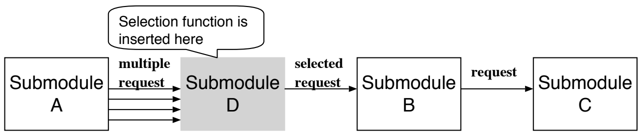


Figure 4: Overview of proposed router architecture.



(a) Outline drawing of selection functions.



(b) Selection functions are implemented into submodule-D.

Figure 5: Insertion of selection functions when adaptive routing is used.

3.4.4 The fully adaptive router

Next, we adopt Duato's protocol [3] as fully adaptive routing. It has an escape path for deadlock avoidance, in which we use west-first routing. Since it is fully adaptive routing, it outperforms west-first routing. When fully adaptive routing is used, however, output channels must send the state of the next router to selection functions to avoid deadlock. Therefore, the delay of RS stages changes as follows:

$$T_{\mathrm{RS}}=G_{\mathrm{rs}}+2W_{\mathrm{segment_d}}+W_{\mathrm{segment_b}}. \tag{14}$$

It requires extra segmented wire delay. The effect is evaluated in Section 4.

4. RESULTS AND DISCUSSION

4.1 Simulation methodology

We design the proposed router models in STMicroelectronics 28-nm FD-SOI process technology to evaluate wire delay, gate delay, and area. They are synthesized by Synopsys Design Compiler, and placed and routed by Synopsys IC Compiler. The packet and flit sizes are 5 flits and 16-bit, respectively.

For the performance evaluation, the baseline and proposed routers are compared on the basis of delay models, gate delay, and wire delay. Overall, there are no functional differences between the baseline and proposed routers as already described in Section 3. Therefore, the performance is basically determined by the critical path delay rather than the number of cycles if pipeline depths are equal.

Table 2: Simulation results of gate delay. Each symbol corresponds to that of delay models.

Symbol	Variations	Delay [ns]
G_{rc}	DOR	0.27
	West-first	0.27
	Duato's protocol	0.27
$G_{\mathrm{fifo_{wr}}}$	–	0.34
G_{rs}	West-first	0.38
	Duato's protocol	0.70
G_{arb}	fixed VA	0.92
	round-robin VA with 8 VCs	1.45
$G_{\mathrm{fifo_{rd}}}$	–	0.18
G_{cb}	–	0.44
G_{buffer}	–	0.21

In addition, the reduction of LT stages can be considered by cycle accurate simulation.

We implement separately each router's function to analyze gate delay on the basis of prescribed delay models. After place-and-route, a static timing analysis is carried out. We measure actual wire delay reported by IC Compiler. We implement a special design to measure wire delay, in which two small macros of an inverter are placed far apart at intervals of optional distance. By sending data from one to the other, wire delay is measured. Since inverters are small, delay in a macro is negligible.

4.2 Critical path delay

Table 2 shows the results of the gate delay based on delay models. G_{rc} is constant regardless of routing algorithm. This is because RC logic only compares the current node with the destination node. Instead, the difference between routing algorithms is found in RS stages. G_{arb} is the longest, and the use of many VCs prolongs it further. $G_{\mathrm{fifo_{wd}}}$, $G_{\mathrm{fifo_{rd}}}$, G_{cb}, and G_{buffer} are constant in all cases. On the basis of these data, the critical path delay is evaluated.

Figure 6 shows the wire delay evaluated by IC Compiler. IC Compiler inserts few repeaters automatically with timing constrains loose. Manhattan distance is defined as the distance of a link considering routers as points. Since a tile measures 1mm wide by 1mm long in our design, Manhattan distance corresponds to real length measured in millimeters. From the figure, repeated wire delay mostly increases linearly. Consequently, we can evaluate segmented wire delay by linear approximation and optimally arrange each submodule. Table 3 shows wire delay of each example topology. Each maximum Manhattan distance can be generalized as described, and the case for a 4x4 mesh is shown.

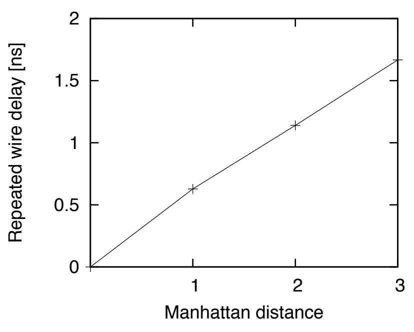


Figure 6: Repeated wire delay vs. Manhattan distance.

Table 3: Wire delay of each example topology.

Example topology	Max. Manhattan distance		Wire delay
	Size: $2^n \times 2^n$	Size: 4×4	Size: 4×4
2D-Mesh	1	1	0.628 ns
Folded Torus	2	2	1.140 ns
Flat. Butterfly	$2^n - 1$	3	1.667 ns

We now finally obtain all numerical data to discuss balanced pipelines and evaluate performance. From here, we do this with respect to each case.

4.2.1 *Case 1: The simple router*

The delay of the baseline router is as follows:

$$T_{\mathrm{RC}} = \max(G_{\mathrm{fifo_{wr}}}, G_{\mathrm{rc}}) = 0.34\,\mathrm{ns}, \quad (15)$$

$$T_{\mathrm{VSA}} = G_{\mathrm{arb}} = 0.92\,\mathrm{ns}, \quad (16)$$

$$T_{\mathrm{ST}} = G_{\mathrm{fifo_{rd}}} + G_{\mathrm{cb}} = 0.62\,\mathrm{ns}, \quad (17)$$

$$T_{\mathrm{LT}} = \begin{cases} 0.63\,\mathrm{ns} & (M=1) \\ 1.14\,\mathrm{ns} & (M=2) \\ 1.67\,\mathrm{ns} & (M=3). \end{cases} \quad (18)$$

M is the maximum Manhattan distance of the topology. Gate delays are not balanced, and VSA is an apparent bottleneck. Looking at wire delay, it becomes the critical path when M becomes two.

Decentralization changes the delay as follows:

$$T_{\mathrm{RC}} = 0.27\,\mathrm{ns} + W_{\mathrm{segment_a}}, \quad (19)$$

$$T_{\mathrm{VSA}} = 0.92\,\mathrm{ns} + W_{\mathrm{segment_b}}, \quad (20)$$

$$T_{\mathrm{ST}} = 0.44\,\mathrm{ns} + W_{\mathrm{segment_c}}, \quad (21)$$

$$T_{\mathrm{data}} \approx 0.21\,\mathrm{ns} + \frac{W_{\mathrm{link}}}{2} = \begin{cases} 0.52\,\mathrm{ns} & (M=1) \\ 0.78\,\mathrm{ns} & (M=2) \\ 1.04\,\mathrm{ns} & (M=3). \end{cases} \quad (22)$$

$W_{\mathrm{segment_a}}$, $W_{\mathrm{segment_b}}$, and $W_{\mathrm{segment_c}}$ appear in T_{RC}, T_{VSA}, and T_{ST}, respectively. Thus no segmented delay appears at more than one pipeline stage. Here we can optimize segmented wire delay to balance pipelines on the basis of liner approximation. Specifically, the following procedure balances pipelines. Note that this method affects only the wire delay rather than the gate delay in contrast to gate sizing and pipeline refactoring.

- **STEP1:** Initially take 0 for every segmented wire delay.
- **STEP2:** Extend $W_{\mathrm{segment_a}}$ until T_{RC} becomes equal to T_{ST}. If $W_{\mathrm{segment_a}}$ reaches W_{link} in the interval, $W_{\mathrm{segment_a}} = W_{\mathrm{link}}$ and the remainder become 0. Then the procedure is completed. Otherwise go to the next step.
- **STEP3:** Extend $W_{\mathrm{segment_c}}$ in the same way. If $W_{\mathrm{segment_c}} + W_{\mathrm{segment_a}}$ reaches W_{link} in the interval, the procedure ends at that point. Otherwise go to the next step.
- **STEP4:** Allocate the remaining wire delay, i.e, $W_{\mathrm{link}} - (W_{\mathrm{segment_a}} + W_{\mathrm{segment_c}})$ to each segmented wire delay equally. As a result, all segmented delays become equal.

If the topology is 2D-mesh ($M = 1$), the procedure ends at Step 1. Consequently, the critical path remains VSA and stays constant, but LT stages vanish even in this case. Meanwhile, if the topology is Folded Torus ($M = 2$), the optimized delay becomes the following.

$$T_{\mathrm{RC}} \approx 0.27\,\mathrm{ns} + 0.653\,\mathrm{ns} = 0.923\,\mathrm{ns}, \quad (23)$$

$$T_{\mathrm{VSA}} \approx 0.92\,\mathrm{ns} + 0.003\,\mathrm{ns} = 0.923\,\mathrm{ns}, \quad (24)$$

$$T_{\mathrm{ST}} \approx 0.44\,\mathrm{ns} + 0.483\,\mathrm{ns} = 0.923\,\mathrm{ns}. \quad (25)$$

In this case, the critical path is improved by 10% compared with the baseline router. Each submodule is arranged at intervals of the same rate as segmented wire delay.

In the case of Flattened butterfly ($M = 3$), the optimized delay becomes as follows.

$$T_{\mathrm{RC}} \approx 0.27\,\mathrm{ns} + 0.83\,\mathrm{ns} = 1.10\,\mathrm{ns}, \quad (26)$$

$$T_{\mathrm{VSA}} \approx 0.92\,\mathrm{ns} + 0.18\,\mathrm{ns} = 1.10\,\mathrm{ns}, \quad (27)$$

$$T_{\mathrm{ST}} \approx 0.44\,\mathrm{ns} + 0.66\,\mathrm{ns} = 1.10\,\mathrm{ns}. \quad (28)$$

The critical path is improved by 34% compared with the baseline router, and we refer to this rate as the improvement rate. We can see from the above that the decentralized router effectively improves the performance as the maximum Manhattan distance increases. After reaching Step 4, the critical path delay increases only at a rate of the third part of that of the baseline router. In that context our proposal expands the availability of low latency topologies.

4.2.2 *Case 2: The router with many VCs*

The use of many VCs drastically lengthens T_{VSA} as follows:

$$T_{\mathrm{VSA}} = G_{\mathrm{arb}} = 1.45\,\mathrm{ns}. \quad (29)$$

The delay of the decentralized router is summarized as follows:

$$T_{\mathrm{RC}} \approx \begin{cases} 0.27\,\mathrm{ns} + 0.63\,\mathrm{ns} = 0.90\,\mathrm{ns} & (M=1) \\ 0.27\,\mathrm{ns} + 1.14\,\mathrm{ns} = 1.41\,\mathrm{ns} & (M=2) \\ 0.27\,\mathrm{ns} + 1.18\,\mathrm{ns} = 1.45\,\mathrm{ns} & (M=3), \end{cases} \quad (30)$$

$$T_{\mathrm{VSA}} \approx \left\{ 1.45\,\mathrm{ns} + 0.00\,\mathrm{ns} = 1.45\,\mathrm{ns} \quad (M \le 3), \right. \quad (31)$$

$$T_{\mathrm{ST}} \approx \begin{cases} 0.44\,\mathrm{ns} + 0.00\,\mathrm{ns} = 0.44\,\mathrm{ns} & (M \le 2) \\ 0.44\,\mathrm{ns} + 0.49\,\mathrm{ns} = 0.93\,\mathrm{ns} & (M=3). \end{cases} \quad (32)$$

In this case T_{VSA} is too large, and hence the critical path of both the baseline and proposed routers are the same when M is less than four. When M is three, the improvement rate becomes 13%. This shows that routers with many VCs are insulated from the influence of link delay.

4.2.3 *Case 3: The partially adaptive router*

West first routing adds RS stages as follows:

$$T_{\mathrm{RS}}{=}G_{\mathrm{rs}} = 0.38\,\mathrm{ns}. \tag{33}$$

The other gate delays do not change and the critical path of the baseline router remains at 0.92 ns. Decentralization changes the delay as follows:

$$T_{\mathrm{RC}}{=}0.27\,\mathrm{ns} + W_{\mathrm{segment_a}}, \tag{34}$$
$$T_{\mathrm{RS}}{=}G_{\mathrm{rs}} = 0.38\,\mathrm{ns} + W_{\mathrm{segment_d}}, \tag{35}$$
$$T_{\mathrm{VSA}}{=}0.92\,\mathrm{ns} + W_{\mathrm{segment_b}}, \tag{36}$$
$$T_{\mathrm{ST}}{=}0.44\,\mathrm{ns} + W_{\mathrm{segment_c}}, \tag{37}$$
$$T_{\mathrm{data}}{\approx}0.21\,\mathrm{ns} + \frac{W_{\mathrm{link}}}{3} = \begin{cases} 0.42\,\mathrm{ns} & (M=1) \\ 0.59\,\mathrm{ns} & (M=2) \\ 0.77\,\mathrm{ns} & (M=3). \end{cases} \tag{38}$$

$W_{\mathrm{segment_d}}$ appears in T_{RS}. Once again, no segmented delay appears in more than one pipeline stage, and hence we can balance pipelines in the same manner as the foregoing procedure. In short, $W_{\mathrm{segment_d}}$ is extended between Step 2 and Step 3.

As a result, the optimized delay for each Manhattan distance becomes as follows:

$$T_{\mathrm{RC}}{\approx}\begin{cases} 0.27\,\mathrm{ns} + 0.63\,\mathrm{ns} = 0.90\,\mathrm{ns} & (M=1) \\ 0.27\,\mathrm{ns} + 0.65\,\mathrm{ns} = 0.92\,\mathrm{ns} & (M\ge 2), \end{cases} \tag{39}$$
$$T_{\mathrm{RS}}{\approx}\begin{cases} 0.38\,\mathrm{ns} + 0.00\,\mathrm{ns} = 0.38\,\mathrm{ns} & (M=1) \\ 0.38\,\mathrm{ns} + 0.49\,\mathrm{ns} = 0.87\,\mathrm{ns} & (M=2) \\ 0.38\,\mathrm{ns} + 0.54\,\mathrm{ns} = 0.92\,\mathrm{ns} & (M=3), \end{cases} \tag{40}$$
$$T_{\mathrm{VSA}}{\approx}\begin{cases} 0.92\,\mathrm{ns} + 0.00\,\mathrm{ns} = 0.92\,\mathrm{ns} & (M\le 3), \end{cases} \tag{41}$$
$$T_{\mathrm{ST}}{\approx}\begin{cases} 0.44\,\mathrm{ns} + 0.00\,\mathrm{ns} = 0.44\,\mathrm{ns} & (M\le 2) \\ 0.44\,\mathrm{ns} + 0.48\,\mathrm{ns} = 0.92\,\mathrm{ns} & (M=3). \end{cases} \tag{42}$$

By adding selection functions, a place to stow wire delay increases. Consequently, $W_{\mathrm{segment_b}}$ remains at 0.92 ns and the critical path is constant if M does not exceed three. The improvement rates are 0%, 19%, and 45% when M is 1, 2, and 3, respectively. By increasing wire delay much further, the critical path delay increases only at a rate of the fourth part of that of the baseline router.

4.2.4 *Case 4: The fully adaptive router*

The use of Duato's protocol prolongs the delay of RS stages as follows:

$$T_{\mathrm{RS}}{=}G_{\mathrm{rs}} = 0.70\,\mathrm{ns}. \tag{43}$$

Besides, T_{RS} of the decentralized router also increases as follows, since the state of the next router is sent backward.

$$T_{\mathrm{RS}}{=}0.70\,\mathrm{ns} + 2W_{\mathrm{segment_d}} + W_{\mathrm{segment_b}}. \tag{44}$$

Here T_{RS} has double $W_{\mathrm{segment_d}}$, and thus it increases rapidly as $W_{\mathrm{segment_d}}$ is extended. Considering this alteration, the optimized procedure is changed in the following respect.

- After $W_{\mathrm{segment_d}}$ reaches G_{VSA}, $W_{\mathrm{segment_d}}$ is never extended.

As for the rest, pipelines are balanced in the same way as the case of west-first routing.

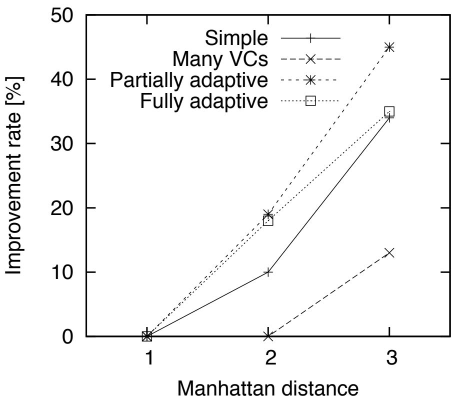


Figure 7: Manhattan distance vs. improvement rate of each case.

The results are as follows:

$$T_{\mathrm{RC}}{\approx}\begin{cases} 0.27\,\mathrm{ns} + 0.63\,\mathrm{ns} = 0.90\,\mathrm{ns} & (M=1) \\ 0.27\,\mathrm{ns} + 0.65\,\mathrm{ns} = 0.92\,\mathrm{ns} & (M=2) \\ 0.27\,\mathrm{ns} + 0.793\,\mathrm{ns} = 1.063\,\mathrm{ns} & (M=3), \end{cases} \tag{45}$$
$$T_{\mathrm{RS}}{\approx}\begin{cases} 0.70\,\mathrm{ns} + 2\times 0.00\,\mathrm{ns} + 0.00\,\mathrm{ns} & \\ = 0.70\,\mathrm{ns} & (M=1) \\ 0.70\,\mathrm{ns} + 2\times 0.11\,\mathrm{ns} + 0.00\,\mathrm{ns} & \\ = 0.92\,\mathrm{ns} & (M=2) \\ 0.70\,\mathrm{ns} + 2\times 0.11\,\mathrm{ns} + 0.14\,\mathrm{ns} & \\ = 1.06\,\mathrm{ns} & (M=3), \end{cases} \tag{46}$$
$$T_{\mathrm{VSA}}{\approx}\begin{cases} 0.92\,\mathrm{ns} + 0.00\,\mathrm{ns} = 0.92\,\mathrm{ns} & (M\le 2) \\ 0.92\,\mathrm{ns} + 0.143\,\mathrm{ns} = 1.063\,\mathrm{ns} & (M=3), \end{cases} \tag{47}$$
$$T_{\mathrm{ST}}{\approx}\begin{cases} 0.44\,\mathrm{ns} + 0.00\,\mathrm{ns} = 0.44\,\mathrm{ns} & (M=1) \\ 0.44\,\mathrm{ns} + 0.38\,\mathrm{ns} = 0.82\,\mathrm{ns} & (M=2) \\ 0.44\,\mathrm{ns} + 0.623\,\mathrm{ns} = 1.063\,\mathrm{ns} & (M=3). \end{cases} \tag{48}$$

The improvement rates are 0%, 18%, and 35% when M is 1, 2, and 3, respectively. They are smaller than in the case of west first routing.

4.2.5 *Summary of the case studies*

Figure 7 shows the improvement rate of each case. The best rate is achieved in the case of the partially adaptive router. There are three reasons for this. First, its pipelines are deep. This means the link delay is split into more segments, and consequently critical path increases slowly when the link delay increases. Second, the differences between the longest stage and the other stages are large. This means the point at which the critical path increases becomes drawn out. Thirdly, it has no signals sent backward, and hence there is no extra segmented delay. All these conditions combine to make decentralization the effective way to improve operating frequency.

On the other hand, other cases do not combine them. The fully adaptive router has signals sent backward. The simple router has fewer pipeline stages than the adaptive router. In the router with many VCs, VSA stages are too long, and hence LT stages do not become critical paths when M is small. The improvement, however, is achieved in all cases and LT stages are certainly deleted, so the effect becomes larger and larger as link delay increases.

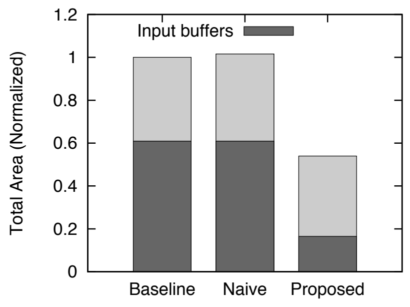


Figure 8: Total area of each router architecture.

4.3 Area budgets

To evaluate area budgets, we compare the baseline, naive decentralized, and proposed routers. They correspond to the first case study, namely the simple case. This evaluation reveals the overhead of naive architecture and the effect of the proposed method. Figure 8 shows the total area of each router architecture. The naive router increases area by 1.6% owing to state machines and additional signals. On the other hand, the proposed router decreases area by 46.0% thanks to the buffer using D latches. This is because input buffers in a router occupy the greater part, *e.g.* 60.9% in this case.

4.4 Discussion

Besides packet switched networks, which this paper deals with, Circuit Switched (CS) networks [7] are proposed. CS networks reserve circuits between two nodes, guaranteeing the full bandwidth of the channel. The CS network and our proposal can coexist as a hybrid design. Hybrid designs are implemented practically in MIT RAW [13] and Tilera TILE64 [5]. Furthermore, a recent study [2] also suggests the hybrid network in which a packet switched network requests channel and a circuit-switched network returns ack and transfer data. Henceforth, both packet switched and circuit switched networks should be studied.

We can make efficient use of the reduction of the critical path for various purposes. A straightforward use is for increasing operating frequency. Our decentralized approach speeds up routers by up to 82%. Alternatively, we can also allow a margin in the frequency/voltage for variability-tolerant and low power designs. Variability-tolerant designs become more and more important as technology evolves. From the above, our decentralized router overcomes the interconnect bottleneck in both delay and energy and increasing variability caused by technology scaling.

5. CONCLUSIONS

We have demonstrated that decentralization of an on-chip router eliminates LT stages, and balanced pipelines improve the critical path by up to 45% in 28-nm process technology. Four case studies have established that our approach is efficient in various routers, and our study provides the framework for future studies about decentralized router architecture. Furthermore, the low-cost decentralized buffer has been proposed. As technology advances, our decentralized routers become more and more beneficial, and hence they are of efficient solutions to avoid the interconnect bottleneck.

Future work will consider new topologies and layouts for decentralized routers. Moreover, future work should discuss router architecture considering both the number of hops and physical delay. Specifically, router performance should be evaluated by the product of the number of hops and critical path delay. Last but not least, our study suggests that decentralized high-speed routers with deep pipelines and low-latency topologies are of efficient solutions in the nano-scale era.

6. ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI S #2522002. The authors also would like to thank to the VDEC Tokyo with Synopsys for EDA tools, and STMicroelectronics for its cooperation.

7. REFERENCES

[1] P. Abad, V. Puente, P. Prieto, and J. A. Gragorio. Rotary router: An efficient architecture for CMP interconnection networks. In *Proc. of the Int'l Symp. on Computer Architecture*, June 2007.

[2] M. Anders. High-performance energy-efficient NoC fabrics. In *Proc. of the Int'l Symp. on Networks-on-Chip*, Sept. 2014.

[3] J. Duato, S. Yalamanchilli, and L. M. Ni. *Interconnection networks: An engineering approach*. Morgan Kaufmann, 2003.

[4] C. J. Glass and L. M. Ni. The turn model for adaptive routing. In *Proc. of the Int'l Symp. on Computer Architecture*, May 1992.

[5] D. W. P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal. On-chip interconnection architecture of the tile processor. *IEEE Micro*, Sept. 2007.

[6] R. Ho, K. W. Mai, and M. A. Horowitz. The future of wires. *Proc. of the IEEE*, Apr. 2001.

[7] N. E. Jerger, L.-S. Peh, and M. Lipasti. Circuit-switched coherence. In *Proc. of the Int'l Symp. on Networks-on-Chip*, Apr. 2008.

[8] J. Kim, J. Balfour, and W. J. Dally. Flattened butterfly topology for on-chip networks. In *Proc. of the Int'l Symp. on Microarchitecture*, Dec. 2007.

[9] G. Michelogiannakis and W. J. Dally. Router designs for elastic buffer on-chip networks. In *Proc. of the Conference for High Performance Computing Networking, Storage and Analysis*, Dec. 2009.

[10] U. Y. Ogras and R. Marculescu. It's a small world after all: NoC performance optimization via long-range link insertion. *IEEE Trans. on VLSI Systems*, July 2006.

[11] A. Roca, C. Hernández, J. Flich, F. Silla, and J. Duato. Silicon-aware distributed switch architecture for on-chip networks. *Journal of Systems Architecture*, Aug. 2013.

[12] I. Seitanidis, A. Psarras, E. Kalligeros, C. Nicopoulos, and G. Dimitrakopoulos. ElastiNoC: A self-testable distributed VC-based network-on-chip architecture. In *Proc. of the Int'l Symp. on Networks-on-Chip*, Sept. 2014.

[13] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwa. The RAW microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, Aug. 2002.

[14] R. Yasudo, T. Kagami, H. Amano, Y. Nakase, M. Watanabe, T. Oishi, T. Shimizu, and T. Nakamura. Design of a low power NoC router using marching memory through type. In *Proc. of the Int'l Symp. on Networks-on-Chip*, Sept. 2014.

[15] B. Zhao, Y. Zhang, and J. Yang. A speculative arbiter design to enable high-frequency many-VC router in NoCs. In *Proc. of the Int'l Symp. on Networks-on-Chip*, Apr. 2013.