PAPER

Two-Level FIFO Buffer Design for Routers in On-Chip Interconnection Networks

Po-Tsang HUANG[†a], ***Student Member and*** **Wei HWANG**[†], ***Nonmember***

SUMMARY The on-chip interconnection network (OCIN) is an integrated solution for system-on-chip (SoC) designs. The buffer architecture and size, however, dominate the performance of OCINs and affect the design of routers. This work analyzes different buffer architectures and uses a data-link two-level FIFO (first-in first-out) buffer architecture to implement high-performance routers. The concepts of shared buffers and multiple accesses for buffers are developed using the two-level FIFO buffer architecture. The proposed two-level FIFO buffer architecture increases the utilities of the storage elements via the centralized buffer organization and reduces the area and power consumption of routers to achieve the same performance achieved by other buffer architectures. Depending on a cycle-accurate simulator, the proposed data-link two-level FIFO buffer can realize performance similar to that of the conventional virtual channels, while using 25% of the buffers. Consequently, the two-level FIFO buffer can achieve about 22% power reduction compared with the similar performance of the conventional virtual channels using UMC 65 nm CMOS technology.

key words: *two-level FIFO buffer, centralized shared buffer, router, on-chip interconnection network*

1. Introduction

System-on-Chip (SoC) designs are an integrated solution for merging processor elements (PEs) or intellectual properties (IPs) in communications, multimedia and consumer electronics. A successful SoC design depends on the availability of methodologies that allow designers to meet two major challenges — the miniaturization of a device and interconnecting features, and ultra-large-scale of circuit integration. Modern SoC designs face a number of problems caused by the communication among multiple PE's. Thus, using an on-chip bus to create a platform is a solution for SoC designs. This on-chip bus platform provides interfaces between multiple processor elements and verification environments [1]. However, the requirements for on-chip communication bandwidth and processor elements are growing continually beyond that which can accommodate standard on-chip buses. Moreover, advanced SoC designs using nano-scale technologies face a number of challenges. First, the shared bus architecture will become a development-critical factor for integration with an increasing number of processor elements. Existing bus architectures and techniques are not scalable, and cannot meet the specific requirements associated with low power and high performance [2]. Second, the interconnect delay across the chip exceeds the average clock period of IP blocks. Thus, the ratio of global interconnect delay to average clock period will continue increasing according to the International Technology Roadmap for Semiconductors (ITRS) [3]. Third, advanced technologies increase the coupling effect for interconnects, such as capacitive and inductive crosstalk noise. The increasing coupling effect aggravates power-delay metrics and degrades signal integrity [4]. Fourth, system design and performance are limited by the complexity of the interconnection between the different modules and blocks with a single clock design [3]. As design complexity continues to increase, a global approach is required to effectively transport data and manage on-chip data communication.

Network-on-chip (NoC) designs were investigated as a method for dealing with a number of challenges caused by the scale and complexity of next generation SoC designs [5]. Furthermore, on-chip interconnection networks (OCINs) provide the micro-architecture and the building blocks for NoCs, including network interfaces, routers and link wires [6], [7]. The generic OCIN is based on a scalable network, which considers all requirements associated with on-chip data communication and traffic. OCINs have a few beneficial characteristics, namely, low communication latency, low energy consumption constraints, and design-time specialization. The motivation in establishing OCINs is to achieve performance using a system communication perspective.

Routers are the essential components of OCINs. Figure 1 shows a generic router architecture, consisting of a set of input buffers, an interconnect matrix, a set of output buffers and control circuitries, including a routing controller, an arbiter and an error detector. The control circuitries serve ancillary tasks and implement some functions of the control

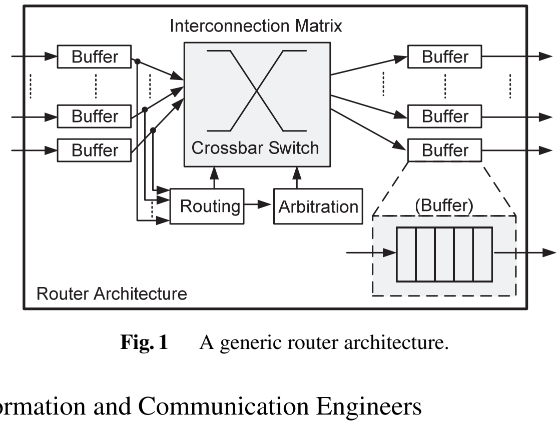


Fig. 1 A generic router architecture.

Manuscript received September 13, 2010.
Manuscript revised June 28, 2011.

[†]The authors are with National Chiao-Tung University, HsinChu 300, Taiwan.
a) E-mail: bug.ee91g@nctu.edu.tw

DOI: 10.1587/transfun.E94.A.2412

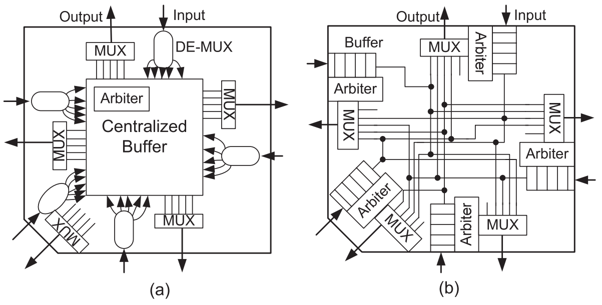


Fig. 2 (a) A router with the centralized buffer (b) A generic router.

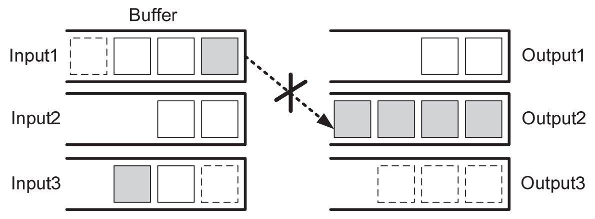


Fig. 3 Head-of-line blocking problem induced by insufficient buffer.

flow protocol. Additionally, the interconnection matrix can be implemented using a single crossbar or by cascading various stages. The control circuitry and interconnection matrix are key components of routers [8]. Furthermore, the buffers significantly increase the overall performance and decrease the complexity of control policies [9], [10]. The buffers allow for local storage of data that cannot be routed immediately. Unfortunately, queuing buffers have high costs in terms of area and power consumption; thus, implementations of OCIN design strive against limited buffer size. In the realm of on-chip buffer design, both size and organization are directly related to performance and power consumption of the OCIN [11]. Buffer size in particular has been thoroughly investigated in [12]–[15]. If a design lacks sufficient buffer space, buffers may fill up too fast to decrease the overall performance; conversely, over-provisioning buffers clearly wastes scarce area resources. Thus, in this work, buffer utilization is optimized via a centralized buffer in a router as shown in Fig. 2(a). Compared with a generic router as shown in Fig. 2(b), a centralized buffer has a shared buffer mechanism allowing channels to share the centralized buffer with sufficient buffer space.

A data-link two-level FIFO (first-in first-out) buffer architecture with the centralized shared buffer is proposed in this paper. The proposed two-level FIFO buffer architecture has a shared buffer mechanism allowing the output channels to share the centralized FIFO with sufficient buffer space. Additionally, the proposed architecture reduces the area and power consumption to achieve the same performance. The remainder of this paper is organized as follows. Section 2 compares and analyzes different buffer architectures and different circuit implementations. The concept of the proposed two-level FIFO buffer architecture is presented in Sect. 3. Section 4 describes the behavior and circuit implementation of the data-link two-level FIFO buffer for the router. The associated two-level FIFO buffer architectures are presented in Sect. 5. Section 6 shows simulation results. Finally, Conclusions are given in Sect. 7.

2. Buffer Implementations and Architectures

The queuing buffer is adopted for routers or network interfaces to store un-routed data. Buffer size and management are directly linked to the flow control policy which affects OCIN performance and resource utilization [8]. Buffer architectures can be classified by their location and circuit implementation of buffers. Queuing buffers consume the most area and power among composing blocks in OCINs [10], [16]. However, insufficient buffer size induces head-of-line blocking problems. Figure 3 shows an example of the head-of-line blocking problem. When head data of a virtual channel cannot be routed and data behind the head data are occupying queuing buffers, network performance is decreased. Nevertheless, head-of-line blocking problems reduce the network performance and increase power consumed during on-chip data communication. Therefore, head-of-line blocking is a key factor when evaluating different buffer architectures.

The buffer circuits can be implemented using registers (flip-flops) or SRAM according to the buffer sizes. For large capacity queuing, the SRAM-based queuing buffer with separated read/write ports is preferred over a register-based buffer [17], [18]. However, SRAM incurs large latency overhead [10]. For achieving high-performance OCINs, register-based buffers are usually realized in the routers with small buffer sizes. Since register-based implementations have a limited capacity due to rapid increasing power consumption and circuit area [11], [16]. In most OCINs, register-based buffers are adopted to provide high bandwidth of on-chip data communication. Consequently, register-based buffers can be classified into four different implementations — (a) Shift Register, (b) Bus-In Shift-Out Register, (c) Bus-In Bus-Out Register, and (d) Bus-In-MUX-Out Register [6].

For the Bus-In register, an arrival packet can be stored in all registers. However, as queuing capacity increases, the driving ability of the sender should be increased for large fan-outs. For the Bus-Out register, all register outputs are connected to a shared output bus via tri-state buffers. The parasitic capacitance of tri-state buffers will increase both delay and power consumption. Therefore, the Bus-In MUX-Out Register with output multiplexers can be utilized to eliminate the parasitic capacitance of tri-state buffers.

Depending on the location of queuing buffers, buffers can be placed before or after the interconnection matrix in a router; these buffers are the input buffer and output buffer, respectively. To be sure, input buffers and output buffers differ. If a data word is delayed in a router with input buffers, it will stall all data words arriving at the same input. None can be processed until the first data word has been forwarded successfully. With output buffers, this situation differs because switching is performed prior to buffering. If a router

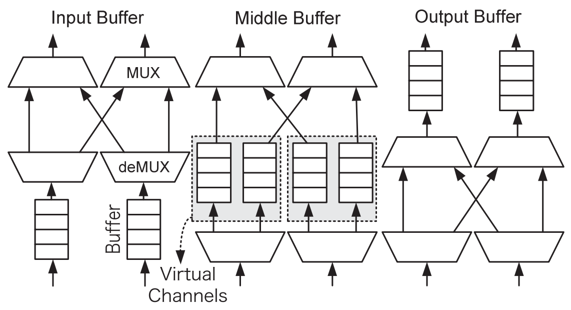


Fig. 4 Diagram of input buffer, middle buffer and output buffer.

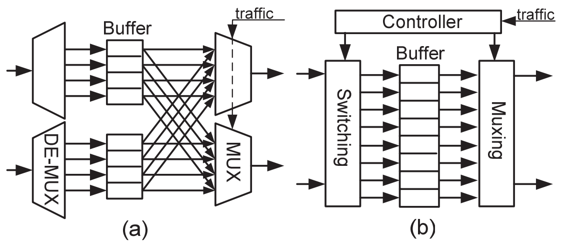


Fig. 5 Concepts of (a) dynamic virtual channel allocation (b) centralized shared buffer.

cannot send data through one of its outputs, the buffers at that output will fill up. However, congestion on outputs has no immediate influence on inputs; that is, successive data words can still be received. An architectural disadvantage of output buffering is that in one cycle, data from multiple input ports may be written to the same output port. Nevertheless, a multiple-access buffer can be implemented in parallel at the output to deal with this shortcoming. Both output buffers and input buffers can cause the head-of-line blocking problem and stall input data. Figure 4 shows the input buffers, middle buffers and output buffers in routers. During middle buffering, the buffer placement moves to the middle of switching circuits. Middle buffer architectures have O(N^2) buffer blocks for an N-port router, while input and output buffering architecture only have O(N) buffer blocks. The middle buffer architecture, however, can reduce the effects of head-of-line blocking via multiple virtual channels during switching. This is a trade-off between traffic problems and buffer sizes.

Since buffer resources are costly in resource-constrained OCIN environments, minimizing buffer size without adversely affecting performance is essential. However, based on observed traffic patterns, buffer size and architecture cannot be changed dynamically during operation. Therefore, some approaches [11], [12] optimize predetermined buffer size during the design stage via a detailed analysis of application-specific traffic patterns. Additionally, static virtual channel allocation techniques were proposed to optimize the performance, area and power for target applications based on the traffic characteristics [14], [19].

For general-purpose and reconfigurable SoC executing different applications, advanced buffer architectures maximize the utilization of buffers under different traffic patterns in NoC applications. As virtual channels are not equally used in different applications, dynamically allocated multiqueue (DAMQ) buffer schemes were proposed to share a common buffer [20]–[23]. However, these approaches are not suited to OCIN implementation, which is typically resource-constrained [24]. Moreover, NoC applications are intolerant of large latency against the quality of service constraint. Hence, in view of resource and latency overhead, dynamic virtual channel allocation schemes were proposed to maximize throughput for resource-constrained OCIN [24]–[28]. Figure 5(a) shows the concept of dynamic virtual channel allocation techniques to share the virtual channels and arbitrate output packets based on the traffic conditions. The dynamic virtual channel regulator (ViChaR) proposed in [24] introduced a unified buffer structure that dynamically allocated virtual channels and buffer resources based on network traffic patterns. The ViChaR has the unified buffer structure and unified control logic. The unified buffer structure shares buffers in virtual channels for each input port. Additionally, the unified control logic controls the arriving/departing pointers and virtual channel allocation of each virtual channel via virtual channel control tables and dispensers. However, the hardware overhead would increase non-linearly. In view of this, other dynamically-allocated virtual channel architectures were proposed by inspecting the physical link state and speculating the packet transferring [25]–[28]. However, when the shared buffers of an input port are full, these approaches do not provide a mechanism for accessing the buffers of other virtual channels at other input ports. Furthermore, the performance of these dynamical virtual channel allocation schemes is also limited due to the resource-constraints of the pointers and virtual channel control tables.

Figure 5(b) shows the centralized shared buffer architecture that maximizes buffer utilization [29]–[31]. Shared buffer architectures are implemented by centralized buffer organizations, which dynamically alter the buffer size for different channels. The input packets from different ports can access all buffers without any head-of-line blocking. This architecture enhances OCIN performance regardless of traffic type. Shared buffering, in addition, achieves the best buffer utility with the fewest memory elements. We have proposed a two-level FIFO buffer architecture to realize the centralized shared buffer via a pointer scheduler [29]. However, the limited size of the pointer scheduler still reduces the performance of the centralized shared buffer. Therefore, other centralized shared buffer architectures enhance the buffer utilization via allocation tables [30], [31]. Nevertheless, the control mechanisms of these shared buffer architectures are more complex than those of other buffer architectures and increase the pipeline stages. Hence, the new proposed data-link two-level FIFO buffer architecture is utilized as the shared buffer architecture to simplify the shared buffer architecture and achieve better performance than other buffer architectures while not increasing buffer size.

3. Concept of Two-Level FIFO Buffer Scheme

The proposed two-level FIFO buffer is constructed by a centralized level-2 FIFO and distributed level-1 FIFOs at output channels. Figure 6 illustrates the data flow of the two-level FIFO buffer scheme. The distributed level-1 FIFOs performs output queues for output channels, and the centralized level-2 FIFO is a unified shared buffer for all output channels. The purposes of distributed level-1 FIFOs is to provide a linear increasing of the FIFO sizes to retrieve the fixed sizes of the centralized level-2 FIFO. The operation of the two-level FIFO buffer is described as follows. After switching packets, the packets are dispensed to the distributed level-1 FIFOs of output channels. If the distributed level-1 FIFO is full or congestion exists in an output channel, packets are dispensed to the centralized level-2 FIFO to prevent head-of-line blocking problems. The centralized level-2 FIFO reduces head-of-line blocking problems via a unified shared buffer to increase the OCIN performance. This unified shared buffer is utilized for all input/output channels that can access all memory elements in the shared buffer. Moreover, the multiple-access mechanism of the shared buffer is also provided for all input/output channels to keep the data flows in OCINs. Therefore, the input/output channels can send/get data to/from the shared buffer at the same time slot via multiple accesses of the shared buffer. Additionally, the centralized level-2 FIFO maximizes buffer utilization. In view of the operation of the two-level FIFO buffer, the arbiter only manages the order of switching packets in output channels.

The centralized level-2 FIFO achieves shared buffering using data-link-based FIFO. Figure 7 presents the concept of the data-link-based FIFO, which takes advantage of data continuity in an FIFO queue. Each slot in the data-link-based FIFO has two stored fields, the data field and linker field. In a slot, the data field stores a flit and linker field stores the address of the next slot, which may not be the adjacent slot in the data-link-based FIFO. In the other words, the linker[i] will store the address of the flit[i+1] in the same FIFO queue. Therefore, the read controller reads the next datum depending on the address stored in the linker field.

The two-level FIFO buffer scheme can be employed at the flit level or packet level depending on flow control techniques, store-and-forward switching, virtual cut-through switching or wormhole switching [32]. Wormhole flow control was proposed to improve performance at the flit level and relaxes the constraints on buffer sizes. Therefore, the wormhole switching technique is the most popular switching technique in packet-switching-based OCINs [33]–[35]. At the flit level, when more than one packet are sent to the same output, the links between these packets cannot be constructed. Therefore, the two-level FIFO buffer needs an extra linker table to record the linked addresses if the tail flit of the front packet is not arrived. Figure 8 gives an example of the two-level FIFO buffer scheme based on a 5input/5output router in a mesh OCIN at the flit level. Therefore, this router is connected to the east router (E), south router (S), west router (W), north router (N), and processor element (P). The flits in the neighbor routers will be dispensed to this router. The first capital letter of a flit indicates the output port of the flit, and the second capital letters (.H, .D and .T) represent the header flit, data flit and tail flit in a packet, respectively. In the two-level FIFO buffer, the first two capital letters indicate the input port and output port of this packet. For example, ES means a packet has an EN turn in this router. In other word, this packet is from the east router, and will be dispensed to the north router. For the output channel S, the packet order is ES–NS–PS–PS, and the centralized level-2 FIFO will construct the links in the

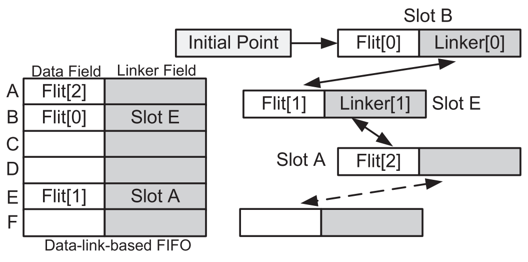


Fig. 7 Concept of the data-link-based FIFO.

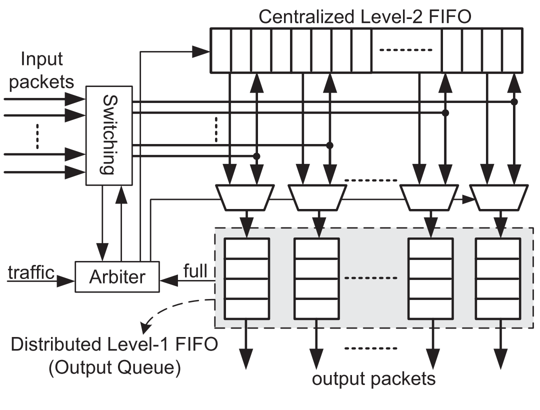


Fig. 6 Data flow of two-level FIFO buffer scheme.

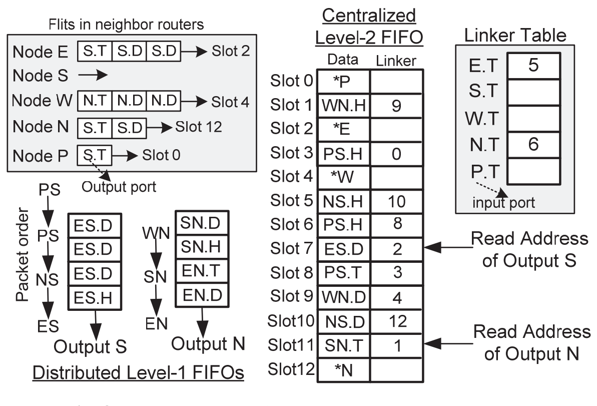


Fig. 8 An example of two-level FIFO buffer scheme.

linker fields based on the packet order. The linker field will store the address of the linked slot. The read address of each output channel denotes the first flit in the centralized level-2 FIFO. When the router N sends a request for this router, the distributed level-1 FIFO will dispense the EN.D flit to the router N, and the flit in the slot 11 will be transferred to the distributed level-1 FIFO. At the same time, the read address of output N will be changed to slot 1. Additionally, the data flit from the router W will be stored into slot 4 that is linked to slot 9. In this example, the packets from E, N and P are routed to the output S, and the order of these packets is E–N–P. The header flit of packet N should be linked to the tail flit of packet E. However, the tail flit of packet E is not dispensed to this router yet. In view of this, the two-level FIFO buffer scheme needs an extra linker table to reconstruct the link by recording the linker for the tail flit.

4. Two-Level FIFO Buffer Architecture

The two-level FIFO buffer architecture is implemented using register-based buffer and consists of a data-link scheduler, distributed level-1 FIFOs, and a data-link-based centralized level-2 FIFO. Figure 9 shows the architecture of the data-link two-level FIFO buffer. The operation of the two-level FIFO buffer router is briefly described as follows. When input packets arrive at the two-level FIFO buffer architecture, the header decoder first de-multiplexes input data from header information. The data-link scheduler then schedules empty buffers and sends de-multiplexed data to the centralized level-2 FIFO. The link scheduler records the address of the output buffer in the linker fields. When acknowledge signals are asserted from the next stage, the distributed level-1 FIFO will transfer output data. Moreover, the data-link scheduler transfers the address, which indicates the bottom of the output buffer, to the centralized level-2 FIFO. The centralized level-2 FIFO delivers accuracy data to the level-1 FIFO. The details of the functional blocks in the two-level FIFO buffer architecture are described as follows.

4.1 Header Decoder and Routing

The packets delivered from processor elements contain headers and payloads. The headers describe the paths the packets will go through. Header information depends on the routing algorithm and OCIN architecture. The two-level FIFO buffer scheme can be employed for both deterministic routing and adaptive routing algorithms.

4.2 Data-Link Scheduler and Centralized Level-2 FIFO

The data-link scheduler and centralized level-2 FIFO are kernel blocks of the two-level FIFO buffer architecture. Figure 10 shows block diagrams of the data-link scheduler and data-link-based centralized level-2 FIFO. The data-link scheduler consists of a write generator, a wordline encoder, a linker table and linker fields that record the addresses of linked data. The centralized level-2 FIFO is constructed using a read controller and data fields. For k flits in the two-level FIFO buffer, the data fields and linker fields are implemented by k slots (words), which can be accessed via write control signals (wordlines). Each slot contains m-bits in the data field and $\log_2(k)$-bits in the linker field. Restated, the width of the linker fields is $\log_2(k)$-bits to record the linked addresses. The width of the data fields is m-bits, and depends on the physical size of a flit.

The data-link scheduler creates links among output channels using the write generator and linker fields. The write generator generates the writing wordlines for the data

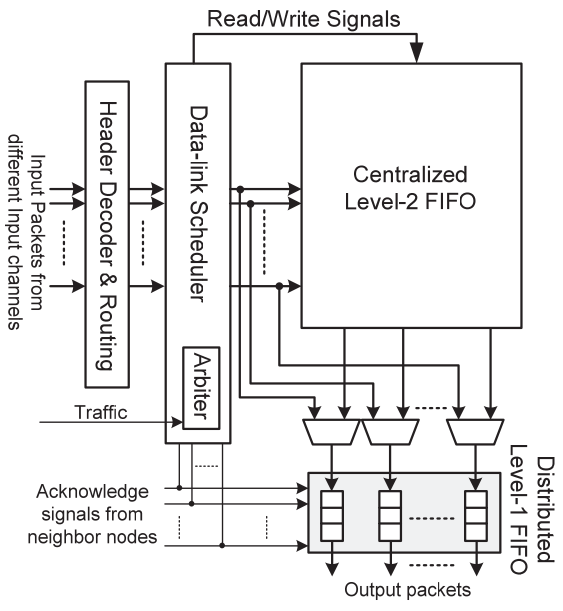


Fig. 9 Two-level FIFO buffer architecture in routers.

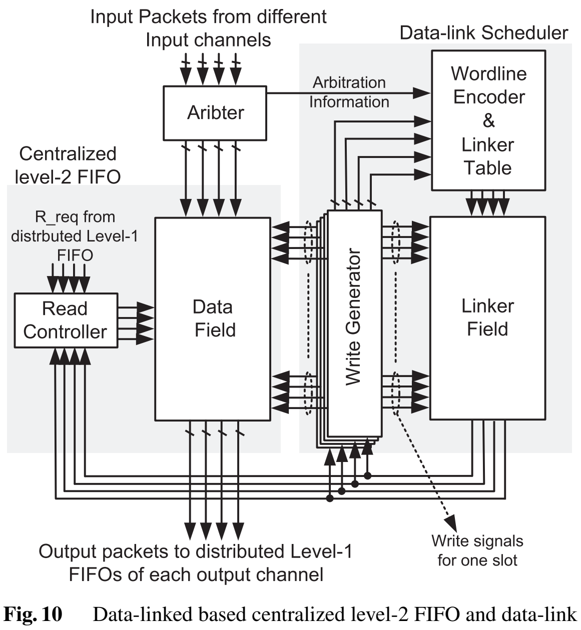

Input Packets from different Input channels
Data-link Scheduler
Aribter
Arbitration Information
Wordline Encoder & Linker Table
Centralized level-2 FIFO
R_req from distrbuted Level-1 FIFO
Read Controller
Data Field
Write Generator
Linker Field
Output packets to distributed Level-1 FIFOs of each output channel
Write signals for one slot

Fig. 10 Data-linked based centralized level-2 FIFO and data-link scheduler.

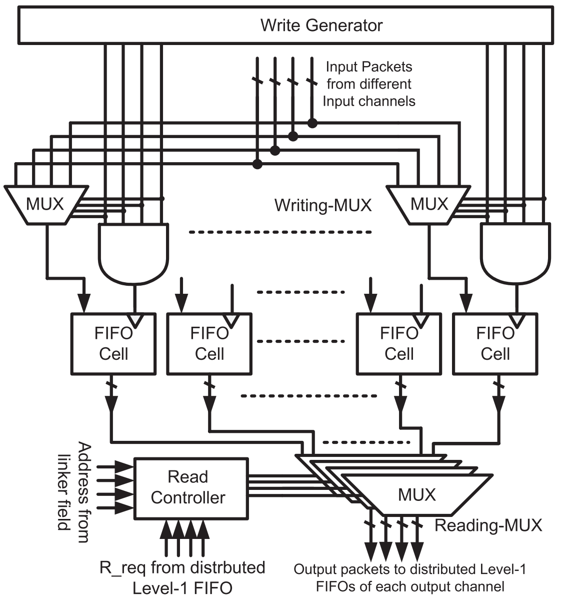

Write Generator
Input Packets from different Input channels
MUX
Writing-MUX
MUX
FIFO Cell
FIFO Cell
FIFO Cell
FIFO Cell
Address from linker field
Read Controller
MUX
Reading-MUX
R_req from distrbuted Level-1 FIFO
Output packets to distributed Level-1 FIFOs of each output channel

Fig. 11 Implementation of the centralized level-2 FIFO.

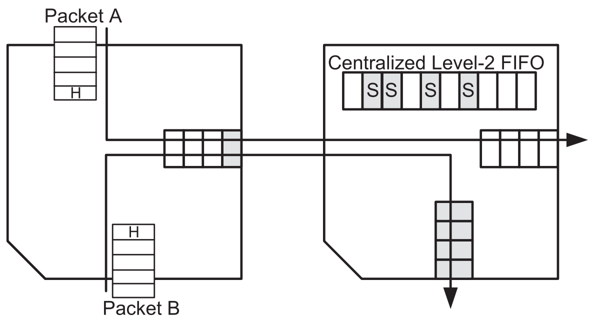


Fig. 12 Example of the arbitration policy in deterministic routing algorithms.

fields to write input flits. While asserting the writing wordlines for the data fields, the linked addresses are produced using the wordline encoder. The wordline encoder encodes these writing wordlines and feeds the encoded addresses (linked addresses) into the linker fields to create links. Therefore, the write generator also latches writing wordlines of the data fields for the linker fields to record the addresses of the next arrival flits. Thus, the switching circuits of the router are utilized in the write generator and data fields based on the link information. Clearly, the read controller obtains addresses from the linker fields to read the next flits of the output channels in the data fields. Hence, the read controller reads the output flits and linked addresses at the same time, and latches the linked addresses for the next transaction. Restated, when the data have been read from the data fields, the read controller records the reading addresses of the data fields to read the next address of the first-in datum in output queues from the linker fields.

The centralized level-2 FIFO provides unified shared buffer and a multiple-access mechanism. In the centralized level-2 FIFO, each slot (word) of the data fields contains m bits to store input data. The linker field is constructed using $\log_2(k)$ bits for storing the addresses of the next datum in queues. Figure 11 shows the schematic of the centralized level-2 FIFO. The data fields and linker fields are both implemented by the Bus-MUX-In MUX-Out registers. For the Bus-MUX-In register, an arrival packet from all possible input channels can be stored in all FIFO cells via the writing MUX, which is designed to select the input channel. The Bus-MUX-In structure provides multiple accesses for the unified shared buffer. Additionally, the Bus-MUX-In structure also performs the switching circuits depending on information from the arbiter and write generator. The arbiter determines the order of packets in an output channel and transfers the routing and arbitration information to the write generator first. Thus, the write generator switches the Bus-In data into appropriate words by writing wordlines and writing MUXs. Further, depending on switching conditions, the write generator transfers writing wordlines to the wordline encoder and creates links. The read controller and reading MUXs decode link addresses and send output data to the distributed level-1 FIFO.

4.3 Distributed Level-1 FIFO

The distributed level-1 FIFOs are designed as output queues located in output channels. Hence, the distributed level-1 FIFOs are implemented using Bus-In Mux-Out registers for shallow output queues. The purpose of distributed level-1 FIFOs is to provide a linear increasing of the FIFO sizes to retrieve the fixed sizes of the centralized level-2 FIFO. Therefore, the size of the distributed level-1 FIFO is usually small, and the Bus-In MUX-Out register is preferred. Moreover, the distributed level-1 FIFOs pre-fetch flits from the centralized level-2 FIFO and to keep the data flow when other output channels are congested.

4.4 Arbiter

The arbiter determines the order of multiple accesses in the same cycle. When more than one packet at different input ports requires the same output port, the arbiter prioritizes the packets. The arbitration algorithm, however, relies on buffer sizes. When buffer size is insufficient, the complexity of the arbiter algorithm increases to eliminate traffic problems. The two-level FIFO buffer architecture provides sufficient buffer sizes using the shared buffer mechanism and multiple accesses for output buffers. That is, the arbiter only decides the order of packets from different input channels when the header flits of these packets arrive at the same time.

The design of the arbiter in the two-level FIFO buffer depends on the characteristic of the routing algorithm. For deterministic routing algorithms, the arbiter can decide the packet order based on the traffic information in the next router. Figure 12 give an example of the arbitration policy. Both packets A and packet B are routed from the left router to the right router. However, the output channel of packet B is congested. If the priorities of packet A and B are at

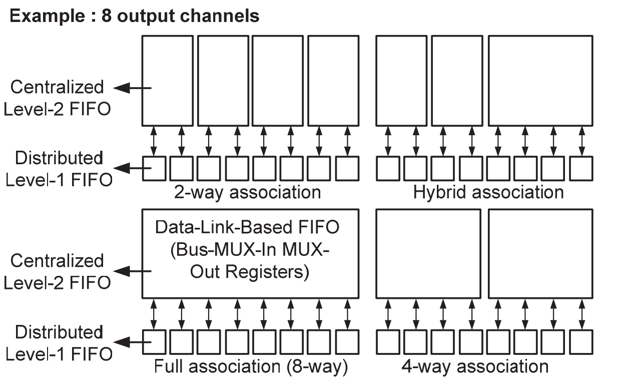


Fig. 13 Different associations between the distributed level-1 FIFO and the centralized level-2 FIFO for 8 output channels.

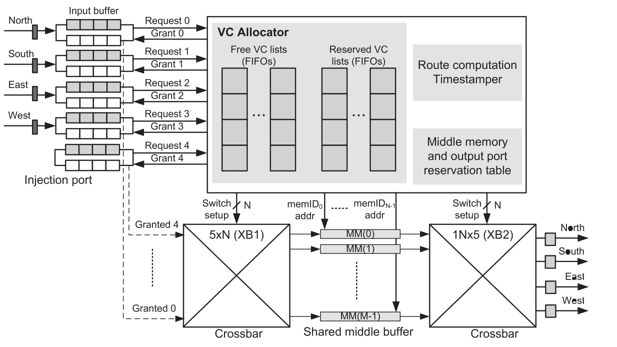


Fig. 14 Router architecture with distributed shared buffer(DSB) [31].

the same level, the order of packet A is in front of packet B. For adaptive routing algorithms, the output of next router cannot be determined in this router. To avoid starvation with low-priority packets and ensure transmission speed of high-priority packets, the two-level FIFO buffer architecture uses the time division multiple access (TDMA) arbitration algorithm, which can be implemented by a counter to transfer priorities for successive input ports. Packet priority determines the position of a packet in the output channel.

5. Associated Two-Level FIFO Buffer Architecture

The two-level FIFO buffer architecture has a unified shared buffer to eliminate head-of-line problems by the data-link-based FIFO and multiple-access mechanism. However, based on the register-based buffer, the power and area overhead of multiple accesses for the centralized level-2 FIFO increases rapidly as the number of output channels increases. Therefore, a trade-off exists between buffer utilities and the power overhead of multiple accesses. That is, the centralized level-2 FIFO can be divided into subgroups for specific output channels. Figure 13 shows the different associations between the distributed level-1 FIFO and centralized level-2 FIFO for 8 output channels. The centralized level-2 FIFO can be deconstructed into different subgroups — two-way association, four-way association, full association or hybrid association. The higher association between the level-2 FIFO and level-1 FIFO will increase buffer utilities of the two-level FIFO buffer. That is, each output channel can access an increased number of buffers in the higher association. Moreover, the physical size of the linker field decreases with the increasing association. Assume the number of available buffers in the centralized level-2 FIFO is k slots. For the m-way association two-level FIFO buffer in an n-port router, the total size of the linker fields is as Eq. (1).

$$\textit{Total Linker Size} = \frac{n}{m}\log_2\left(\frac{k}{n/m}\right) \tag{1}$$

6. Simulation Results

In this section, a cycle-driven simulator is used to evaluate different buffer architectures in SystemC, including output buffer, middle buffer, ViChaR [24], distributed shared buffer (DSB) [31] and the proposed two-level FIFO buffer. The middle buffer architecture establishes multiple virtual channels during switching to reduce head-of-line problems via static virtual channel allocation. The ViChaR architecture provides unified buffer structures at input ports as dynamical virtual channels. Rather than buffering data at the output ports, a DSB router as shown in Fig. 14 uses two crossbar stages with buffering sandwiched in between. Incoming packets are assigned to one of the middle memory buffers with two constraints. First, incoming packets that are arriving at the same time must be assigned to different buffers. Second, an incoming packet cannot be assigned to a buffer that already holds a packet with the same departure time. Additionally, different buffer architectures are also evaluated with different routing algorithms, including XY routing, DyXY [36] and an adaptive routing [37]. Moreover, the proposed data-linked two-level FIFO buffer is implemented using UMC 65 nm standard performance CMOS technology to demonstrate the power consumption and area.

The number of pipeline stages in a router depends on the buffer architecture. Figure 15 presents the pipeline stages of different buffer architectures, and the link traversal (LT) stage indicates flits traverse the link wires to arrive at the downstream router. The middle buffer and ViChaR are realized in 4-stage pipeline routers comprising router computing (RC), virtual channel allocation (VC), switching allocation (SA) and switch traversal (ST). The difference between the middle buffer and ViChaR is in the VC stage. The output buffer router also realizes a 4-stage pipeline consisting of RC, arbitration, SA and ST. The DSB provides a centralized shared buffer to increase performance, and an extra pipeline stage is added into the DSB router to provide high throughput (3 GHz claimed in [31] if using Intel's 65 nm process). Therefore, the DSB router has a five-stage pipeline comprising RC, timestamping (TS), conflict resolution (CR) and VA, first switching traversal (ST1) and middle memory writing (MM_WR), and middle memory read-

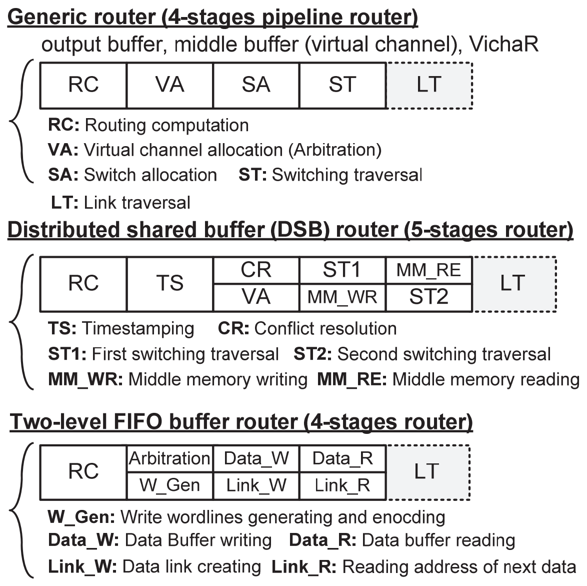

Generic router (4-stages pipeline router)
output buffer, middle buffer (virtual channel), VichaR
RC
VA
SA
ST
LT
RC: Routing computation
VA: Virtual channel allocation (Arbitration)
SA: Switch allocation ST: Switching traversal
LT: Link traversal
Distributed shared buffer (DSB) router (5-stages router)
RC
TS
CR
VA
ST1
MM_WR
MM_RE
ST2
LT
TS: Timestamping CR: Conflict resolution
ST1: First switching traversal ST2: Second switching traversal
MM_WR: Middle memory writing MM_RE: Middle memory reading
Two-level FIFO buffer router (4-stages router)
RC
Arbitration
W_Gen
Data_W
Link_W
Data_R
Link_R
LT
W_Gen: Write wordlines generating and enocding
Data_W: Data Buffer writing Data_R: Data buffer reading
Link_W: Data link creating Link_R: Reading address of next data

Fig. 15 Pipeline stages of the generic router, DSB router and two-level FIFO buffer router.

ing (MM_RD) and second switching traversal (ST2). However, the delay of link wires within two routers (LT stage) dominates the operation frequency of the network while the frequency of network is increased up to 1 GHz and the length of link wires is larger than 2 mm [38]. Therefore, the proposed data-link two-level FIFO buffer also provides a centralized shared buffer without inserting extra pipeline stage, but lower frequency. The 4 pipelining stages include RC, arbitration and write wordlines generating & encoding (W_Gen), data buffer writing (Data_W) and data link creating (Link_W), and data buffer reading (Data_R) and linker reading (Link_R). Moreover, the switching circuit is concealed in write wordlines generating and data link creating as described in Sect. 4.

6.1 Performance Evaluation

According to the cycle-driven simulation in SystemC, Fig. 16 shows the performance of output buffer, middle buffer, ViChaR, DSB and two-level FIFO buffers (including 2-3 hybrid association and full association) with different buffer sizes. The simulation environment is an 8×8 mesh network with an X-Y routing algorithm and uniform traffic patterns. Each packet contains 2, 4 or 8 flits randomly. In a mesh network, each router has 5 inputs and 5 outputs. In a ViChaR router, each input channel has a unified buffer structure with the same buffer size of the input/output buffer. For the middle buffer, each input port has 4 virtual channels. Therefore, the depth of each virtual channel is from 2-flit to 16-flit in this simulation. For the two-level FIFO buffers, the distributed level-1FIFO is set as 2-flit for each output channel. The 2-3 hybrid associated two-level FIFO buffer divides the centralized level-2 FIFO into two subgroups. One is shared by the east and west ports, and the other is shared by the north port, south port and processor element. Fig-

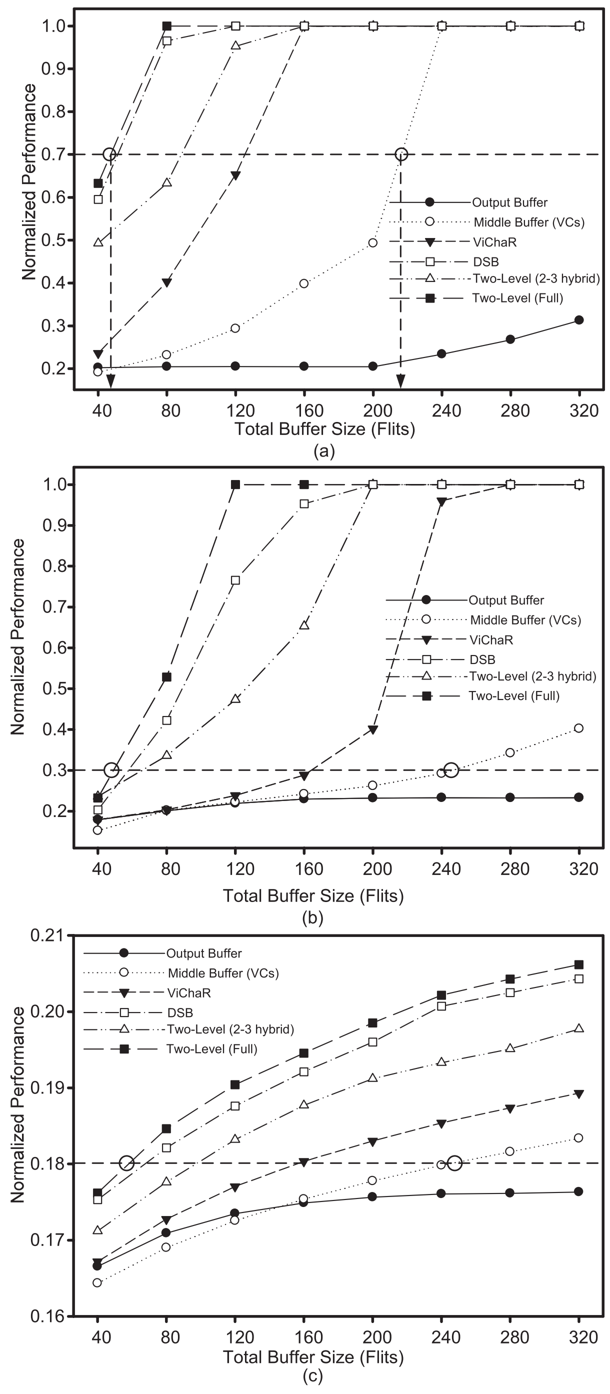


Fig. 16 Normalized performance versus FIFO sizes with different buffer organizations in (a) low injection load (b) medium injection load (c) high injection load.

ures 16(a)–(c) show the simulation results under different injection loads of low (0.15), medium (0.25) and high (0.35), respectively. Performance is normalized to the throughput of infinite buffers within constant cycles under the same injection load. Therefore, the required buffer sizes with different buffer organizations can be easily compared under the same performance. For example, when the normalized performance is 0.7 under the low injection load, the required buffer sizes of the two-level FIFO buffer and middle buffer are 40 flits and 200 flits, respectively. The two-level FIFO buffer architecture performs best with the same size of other buffer architectures regardless of injection loads. For the ViChaR, the unified buffer structure shares buffers in virtual channels for each input port. Thus, the unified control logic controls arriving/departing pointers and virtual channel allocation of each virtual channel through virtual channel control tables and dispensers. However, when the shared buffer of an input port is full, the ViChaR does not provide a mechanism for accessing buffers of other virtual channels in other input ports. For the centralized shared buffer, the performance of the DSB buffer is similar to that of the fully associated two-level FIFO buffer. Moreover, the operation frequency of the DSB buffer is higher than the two-level FIFO buffer due to an extra pipeline stage. Additionally, when the total buffer size is small, the performance of the middle buffer is worse than that of the output buffer. The reason of this phenomenon is due to shallow virtual channels. In high injection load, the throughput of different buffer architectures is quite smaller than that of infinite buffers because the performance is dominated the heavy congestion of the network. Compared to the traditional router with middle buffers, the total buffer size of the two-level FIFO buffer can be reduced to 20%–25% for achieving the same performance.

Different buffer architectures are evaluated by another metric of network performance, namely average latency. Figures 17(a) and 17(b) present the average latencies of different buffer architectures with uniform patterns and hotspot patterns, respectively. The simulation environment is an 8×8 mesh network with an X-Y routing algorithm, and each packet contains 4 or 8 flits randomly. In addition, the total buffer size is set as 160 flits. In hotspot traffic, uniform traffic is applied, but then 30% of the packets change their destination to one of six nodes $(2, 3)$, $(2, 4)$, $(3, 3)$, $(3, 4)$, $(6, 5)$, $(6, 6)$ equally. Compared with the conventional output buffer and middle buffer, the ViChaR, DSB and two-level FIFO buffer architectures can achieve the lower average latencies no matter what the injection load is. In lower injection load, the average latencies of the DSB buffer are larger than those of ViCharR and two-level FIFO buffer because of inserting one extra pipelining stage. With the increasing injection load, the average latencies of the DSB buffer are reaching those of two-level FIFO buffer because the latencies are dominated the heavy congestion of the network. Moreover, the fully associated two-level FIFO buffer can realize the lowest average latencies compared to other buffers. Nevertheless, the boundaries of shared buffers, including ViChaR and two-level FIFO buffer, decrease significantly in hotspot patterns. Restated, the shared mechanism cannot lighten the traffic with hotspots efficiently.

After comparing the buffer architectures of different routing algorithms, Fig. 18 presents the average latency under different routing algorithms with the middle buffer and fully associated two-level FIFO buffer architectures. The routing algorithms are XY routing and DyXY [36] routing

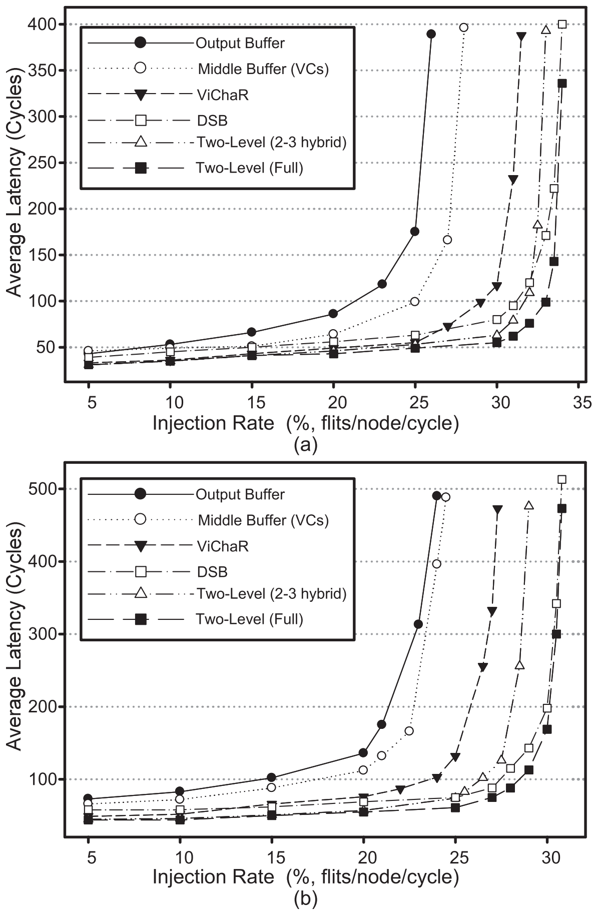


Fig. 17 Average latencies of different buffer architectures in (a) uniform patterns (b) hotspot patterns.

algorithms and an adaptive congestion-aware routing algorithm [37]. In DyXY and adaptive routing algorithms, the two-level FIFO buffer uses the TDMA arbiter described in Sect. 4. These graphs follow the same trend as the latency simulations. Figures 18(a) and 18(b) show the average latencies with uniform random patterns and 6 hotspots in the center region, respectively. The two-level FIFO buffer reduces the influence of performance on average latencies induced by the routing algorithms. In addition, the DyXY routing algorithm with a two-level FIFO buffer performs better than the adaptive algorithm with a two-level FIFO buffer. Moreover, the adaptive routing algorithm increases the average latencies when the injected load is low. However, the adaptive algorithm can achieve the lowest average latencies with high injected load in hotspot patterns.

6.2 Area/Power Analysis

The two-level FIFO buffer architecture is implemented via Synopsys Design Compiler and PrimePower to estimate the area and power consumption based on UMC 65 nm CMOS technology at 1.0 V. Table 1 lists the area, power consumption and maximum operation frequency of routers with different buffer architectures for similar buffer sizes. The

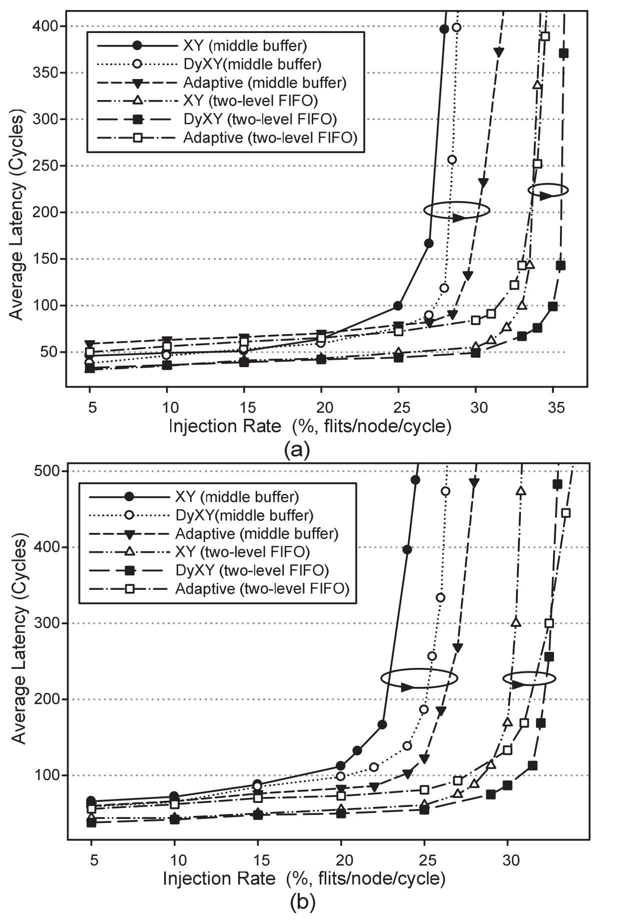


Fig. 18 Average latencies of XY, DyXY and adaptive routing algorithms in (a) uniform patterns (b) hotspot patterns.

Table 1 Area and power comparisons between different buffer architectures in the same buffer size.

Buffer architecture	Area (Mim^2)	Power (mw)	Max Freq.
Middle buffer (Bus-in MUX-out, 160 flits)	60524.3(0%)	16.94(0%)	1.25 GHz
Middle buffer (Bus-in Bus-out, 160 flits)	56832.3(-6.1%)	18.61(+9.9%)	1.33 GHz
ViChaR (160 flits)	79322.6(31.1%)	23.37(+30.0%)	1.11 GHz
DSB (160 flits)	88235.3(+45.7%)	25.31(+49.4%)	1.25 GHz
Fully associated two-level FIFO buffer (158 flits)	83654.8(+38.2%)	23.91(+41.1%)	1 GHz
2-3 hybrid associated two-level FIFO buffer (64 flits for each subgroup, 158 flits)	79053.8(+30.6%)	21.54(+27.2%)	1.11 GHz

power consumption is simulated at 1 GHz under low injection load (0.15). These buffer architectures include the middle buffer using the static virtual channel allocation, a dynamic virtual channel regulator (ViChaR), DSB and two-level FIFO buffer architectures. The middle buffer architectures are implemented as the Bus-In MUX-Out and Bus-In Bus-Out registers, respectively. The middle buffer is implemented as 5 input ports with 4 virtual channels for the input port; each virtual channel has 8 flits and each flit size is 64 bits. Therefore, the total number of flits for each router is 160 flits ($5 \times 4 \times 8$). The ViChaR has a unified buffer structure that dynamically allocates virtual channels and buffer resources according to network traffic patterns. The ViChaR is composed of a unified buffer structure and unified control logics, which are the arriving/departing pointers and the VC control table. In each input port, the size of the unified buffer structure is 32 flits. For fully associated two-level FIFO buffer architecture, the centralized level-2 FIFO is implemented as Bus-MUX-in MUX-out registers; the centralized level-2 FIFO is 128 flits (words) deep and 64 bits wide. For the 2-3 hybrid associated centralized level-2 FIFO, the depth is 64 flits (words) and width is 64 bits for each subgroup. In order to obtain the same size of two-level FIFO buffers for different buffer architectures, each distributed level-1 FIFO has 6 flits with 64 bits that linearly increase FIFO sizes to determine the fixed sizes of the centralized level-2 FIFO. Therefore, the total number of flits in the fully associated and 2-3 hybrid association is 158 flits ($128+6\times5$, $64 \times 2 + 6 \times 5$). The maximum operation frequency of the two-level FIFO buffer is 1 GHz that is smaller than those of the conventional buffer architectures. The DSB can achieve the same frequency as that of the middle buffer via an extra pipeline stage.

For a similar number of flits, the DSB occupies the largest area compared with those of other buffer architectures because of two switch circuits, complex arbitration and a great number of lookup tables. Furthermore, Fig. 19 presents the breakdown of area and power consumption of the two-level FIFO buffer and DSB. The proposed two-level FIFO buffer architecture induces 38.2% area overhead due to multiple accesses of the centralized level-2 FIFO. Nevertheless, the two-level FIFO buffer architecture dissipates the smaller power than the DSB and ViChaR because the

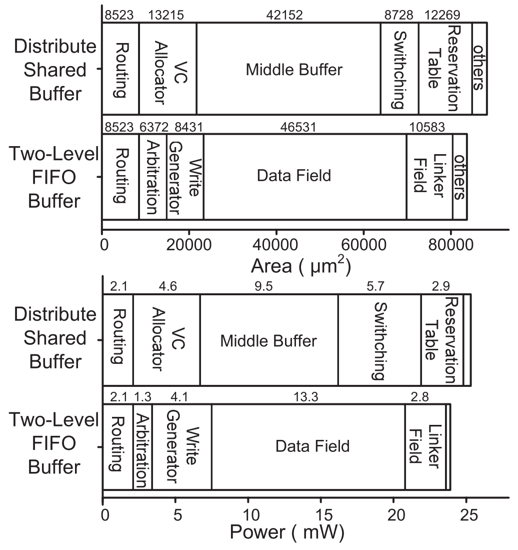


Fig. 19 Breakdown of area and power consumption of two-level FIFO buffer and DSB.

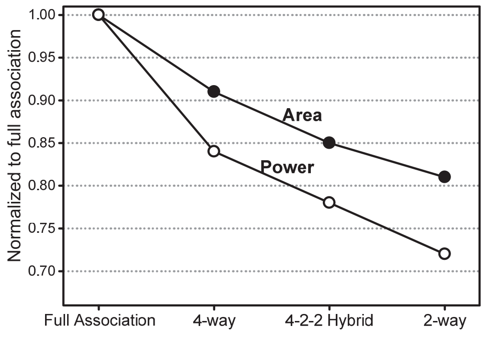


Fig. 20 Power and area analysis of the different associated two-level FIFO buffers in an 8input/8output router.

Table 2 Area and power comparisons between different buffer architectures under similar performance.

Buffer architecture	Area (μm^2)	Power (mw)
Middle buffer (Bus-in MUX-out, 160 flits)	60524.3(0%)	16.94(0%)
ViChaR (80 flits)	52213.6(-13.7%)	15.37(-9.3%)
DSB (40 flits)	47771.6(-21.1%)	13.69(-19.2%)
Fully associated two-level FIFO buffer (40 flits)	44251.3(-26.9%)	13.17(-22.3%)

VC control table dissipates more power than the data-link scheduler.

Although the size of the linked field in the hybrid associated two-level FIFO buffer is larger than that in the fully associated two-level FIFO buffer, the hybrid association uses less power and area overhead by reducing the number of multiple accesses. This is a trade-off between performance and power consumption. Therefore, Fig. 20 presents the power and area analysis of different associated two-level FIFO buffers corresponding to an 8input/8output router. Both the power and area are reduced when the association decreases. With the decreasing association, the number of multiple accesses in each sub-group also decreases but the size of linker fields increases. Therefore, the power and area overheads of the linker fields are both smaller than those of the multiple-access mechanism.

The proposed fully associated two-level FIFO buffer can achieve performance similar to that of the conventional virtual channels, while using 20–25% of the buffers. Therefore, the area and power consumption of the middle buffer, ViChaR, DSB and two-level FIFO buffer are also analyzed under similar performance as listed in Table 2. The power consumption is simulated at 1 GHz under low injection load (0.15). The ViChaR uses half buffers (80 flits) to realize the similar performance. Consequently, the DSB and two-level FIFO buffer achieve the similar performance using one-fourth buffers (40 flits), and each flit size is 64 bits. Based on UMC 65 nm CMOS technology at 1.0 V and 1 GHz, the ViChaR, DSB and proposed two-level FIFO buffer can achieve 9.3%, 19.2% and 22.3% power reduction, respectively.

7. Conclusions

On-chip interconnection network (OCIN) designs have been considered as an effective solution to integrate process-independent interconnection architectures and multi-core systems. Additionally, OCIN performance is directly related to the buffer sizes and utilization. In this paper, a data-link two-level FIFO buffer architecture is presented as a good solution for routers in OCINs based on a shared buffer mechanism and multiple accesses. Additionally, the centralized level-2 FIFO is realized via a data-link scheduler. This buffer architecture with a small buffer size reduces the magnitude of head-of-line blocking problems and performs well. According to the cycle-accurate simulator, the two-level FIFO buffer can realize performance similar to that of the conventional virtual channels, while using 20–25% of the buffers. Based on UMC 65 nm CMOS technology, the proposed data-link two-level FIFO buffer can achieve about 22% power reduction compared with the similar performance of the conventional virtual channels. The two-level FIFO buffer architecture is very useful as alternative design that increases the performance of routers in OCINs.

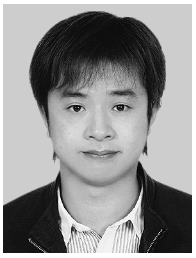
Acknowledgments

This research is supported by National Science Council, R.O.C., under project NSC 98-2220-E-009-026, NSC 98-2220-E-009-027. This work also supported by Ministry of Economic Affairs, R.O.C. The authors would like to thank ITRI, and TSMC for their support.

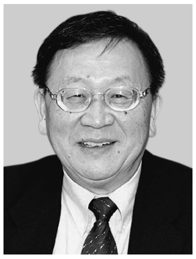
References

[1] V. Chandra, A. Xu, H. Schmit, and L. Pileggi, "An interconnect channel design methodology for high performance integrated circuits," Proc. Des. Autom. Test Eur. Conf. Exhib. (DATE), vol.2, pp.1138–1143, March 2004.
[2] K.-C. Chang, J.-S. Shen, and T.-F. Chen, "Evaluation and design trade-offs between circuit switched and packet switched NOCs for application-specific SOCs," Proc. Des. Autom. Conf. (DAC), pp.143–148, June 2006.
[3] (2005–2009) International Technology Roadmap for Semiconductors. Semiconductor Industry Assoc. [Online]. Available: http://public.itrs.net
[4] P.-T. Huang, W.-L. Fang, Y.-L. Wang, and W. Hwang, "Low power and reliable interconnection with self-corrected green coding scheme for network-on-chip," Proc. IEEE Int. Symp. Netwrok-on-Chip, pp.77–83, April 2008.
[5] L. Benini and G. De-Micheli, "Networks on chips: A new SoC paradigm," Computer, vol.35, no.1, pp.70–78, Jan. 2002.
[6] L. Benini and G. De Micheli, Network on Chips: Technology and Tools, Morgan Kaufmann, 2006.
[7] W.J. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.
[8] Y. Qian, Z. Lu, W. Dou, and Q. Dou, "Analyzing credit-based router-to-router flow control for on-chip networks," IEICE Trans. Electron., vol.E92-C, no.10, pp.1276–1283, Oct. 2009.
[9] R. Beidas and Z. Jianwen, "A queuing-theoretic performance model for context-flow system-on-chip platforms," Proc. Workshop of Embedded Syst. for Real-Time Multimedia, pp.21–26, 2004.

[10] Y. Qian, Z. Lu, and W. Dou, "Worst-case flit and packet delay bounds in workhole networks on chip," IEICE Trans. Fundamentals, vol.E92-A, no.12, pp.3211–3220, Dec. 2009.
[11] J. Kim, "Low-cost router microarchitecture for on-chip networks," Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO-42), pp.255–266, 2009.
[12] G. Varatkar and R. Marculescu, "Traffic analysis for on-chip networks design of multimedia applications," Proc. Des. Autom. Conf. (DAC), pp.795–800, June 2002.
[13] J. Hu, U.Y. Ogras, and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," Proc. IEEE/ACM Int. Conf. of Comput.-Aided Des., pp.354–361, 2004.
[14] J. Hu, U.Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific network-on-chip router design," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.25, no.12, pp.2919–2933, Dec. 2007.
[15] M. Coenen, S. Murali, A. Radulescu, and K. Goossens, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," Proc. IEEE Int. Conf. Hardware/Software Codesign and Syst. Synthesis, pp.130–135, 2006.
[16] W.J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection network," Proc. Des. Autom. Conf. (DAC), pp.684–689, June 2001.
[17] A.K. Kodi, A. Sarathy, and A. Louri, "iDEAL: Inter-router dual-function energy and area-efficient links for network-on-chip (NoC)," Proc. Int. Symp. Comput. Architecture (ISCA), pp.241–250, 2008.
[18] D. Kim, K. Kim, J.-Y. Kim, S. Lee, and H.-J. Yoo, "Memory-centric network-on-chip for power efficient execution of task-level pipeline on a multi-core processor," IET Comput. & Digit. Tech., vol.3, no.5, pp.513–524, 2009.
[19] T.-C. Huang, U.Y. Ogras, and R. Marculescu, "Virtual channels planning for network-on-chip," Proc. Int. Symp. Quality Electron. Des., pp.879–884, 2007.
[20] J. Park, B.W. Okrafka, S. Vassiliadis, and J. Delgado-Frias, "Deign and evaluation of a DAMQ multiprocessor network with self-compacting buffers," Proc. Supercomput., pp.713–722, 1994.
[21] N. Ni, M. Pirvu, and L. Bhuyan, "Circular buffered switch design with wormhole routing and virtual channels," Proc. IEEE Int. Conf. Comput. Des., pp.466–473, 1998.
[22] J. Liu and J.G. Delgado-Frias, "A shared self-compacting buffer for network-on-chip systems," Proc. IEEE Int. Midwest Symp. Circuits Syst., pp.26–30, 2006.
[23] M.A.J. Jamali and A. Khademzadeh, "A new method for improving the performance of network on chip using DAMQ buffer schemes," Proc. Int. Conf. Application Information Comm. Tech., pp.1–6, 2009.
[24] C.A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M.S. Yousif, and C.R. Das, "ViChaR: A dynamic virtual channel regulator for network-on-chip routers," Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO-39), pp.333–346, 2006.
[25] M. Lai, Z. Wang, L. Gao, H. Lu, and K. Dai, "A dynamically-allocated virtual channel architecture with congestion awareness for on-chip routers," Proc. Des. Autom. Conf. (DAC), pp.630–633, June 2008.
[26] M. Lai, L. Gao, W. Shi, and Z. Wang, "Escaping from blocking a dynamic virtual channel for pipelined routers," Proc. Int. Conf. Complex, Intelligent and Software Intensive Syst., pp.795–800, 2008.
[27] M.H. Neishaburi and Z. Zilic, "Reliability aware NoC router architecture using input channel buffer sharing," Proc. 19th ACM Great Lakes Symp. on VLSI, pp.511–516, 2009.
[28] A.K. Kodi, A. Sarathy, A. Louri, and J. Wang, "Adaptive inter-router links for low-power, area-efficient and reliable network-on-chip (NoC) architectures," Proc. Asia and South Pacific Des. Autom. Conf., pp.1–6, 2009.
[29] P.-T. Huang and W. Hwang, "2-Level FIFO architecture design for switch fabrics in network-on-chip," Proc. IEEE Int. Symp. Circuits and Systems, pp.4863–4866, May 2006.
[30] L. Wang, J. Zhang, X. Yang, and D. Wen, "Router with centralized buffer for network-on-chip," Proc. 19th ACM Great Lakes Symp. on VLSI, pp.469–474, 2009.
[31] R.S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Design of a high-throughput distributed shared-buffer NoC router," Proc. IEEE Int. Symp. on Network-on-Chip, pp.69–78, 2010.
[32] W.J. Dally and C.L. Seitz, "The torus routing chip," J. Distributed Computing, vol.3, no.4, pp.267–286, 1979.
[33] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-Tile sub-100-W TeraFLOPS processor in 65-nm CMOS," IEEE J. Solid-State Circuits, vol.43, no.1, pp.29–41, Jan. 2008.
[34] K.-M. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high performance SoC design," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.14, no.2, pp.148–160, Feb. 2006.
[35] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De1, R.V.D. Wijngaart, and T. Mattson, "A 48-Core IA-32 message-passing processor with DVFS in 45 nm CMOS," Proc. IEEE Int. Solid-State Circuits Conf., pp.108–110, Feb. 2010.
[36] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY — A proximity congestion-aware deadlock-free dynamic routing method for network on chip," Proc. Des. Autom. Conf. (DAC), pp.849–852, June 2006.
[37] P.-T. Huang and W. Hwang, "An adaptive congestion-aware routing algorithm for mesh network-on-chip platform," Proc. IEEE System-on-Chip Conf. (SOCC), pp.375–378, Sept. 2009.
[38] C. Hernandez, F. Silla, and J. Duato, "A methodology for the characterization of process variation in NoC links," Proc. Des. Autom. Test Eur. Conf. Exhib. (DATE), pp.685–690, 2010.



Po-Tsang Huang is a student member of the IEEE. He received his B.S. degree from the Department of Electronics Engineering, National Chiao-Tung University, Taiwan, in 2002. He is a Ph.D. student at the Institute of Electronics Engineering, National Chiao-Tung University. His research interests focus on memory system design and low power SoC design with particular emphasis on on-chip interconnection network platform.



Wei Hwang (Fellow IEEE 2001) received the B.Sc. degree from National Cheng Kung University, Taiwan, the M.Sc. degree from National Chiao Tung University, Taiwan and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Manitoba, Winnipeg, MB, Canada, in 1970 and 1974, respectively. From 1975 to 1978, he was an Assistant Professor with the department of Electrical Engineering, Concordia University, Montreal, QC, Canada. From 1979 to 1984, he was an Associate Professor with the department of Electrical Engineering, Columbia University, New York, NY, USA. From 1984 to 2002, he was a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. In 2002, he joined National Chiao Tung University (NCTU), Hsinchu, Taiwan, as the Director of Microelectronics and Information Systems Research Center until 2008, where he currently holds a Chair Professor with the Department of Electronics Engineering. During 2003 to 2007, he served as Co-principal Investigator of National System-on-Chip (NSoC) Program, Taiwan. From 2005 to 2007, he also served as a Senior Vice President and Acting President of NCTU, respectively. He is the coauthor of the book "Electrical Transports in Solids-With Particular Reference to Organic Semiconductors", Pergamon Press, 1981, which has been translated into Russian and Chinese. He has authored or coauthored over 200 technical papers in renowned international journals and conferences, and holds over 150 international patents (including 65 U.S. patents). Prof. Hwang was a recipient of several IBM Awards, including 16 IBM Invention Plateau Invention Achievement Awards, 4 IBM Research Division Technical Awards, and was named an IBM Master Inventor. He was also a recipient of the CIEE Outstanding Electrical Engineering Professor Award in 2004 and Outstanding Scholar Award from the Foundation for the advancement of Outstanding Scholarship for 2005 to 2010. He has served several times in the Technical Program Committee of the ISLPED, SOCC, and A-SSCC. He has also served as the General Chair of 2007 IEEE SoC Conference (SOCC 2007) and 2007 IEEE International Workshop on Memory Technology, Design and Testing (MTDT 2007). Currently, he is serving as Founding Director of Center of Advanced Information Systems and Electronics Research (CAISER) of University System of Taiwan, the Director of ITRI and NCTU Joint Research Center, and a Supervisor of IEEE Taipei Section. He is a Life Fellow of the IEEE.