Efficient Switches for Network-on-Chip Based Embedded Systems

Hsin-Chou Chi and Chia-Ming Wu

Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien, Taiwan

E-mail: hcchi@mail.ndhu.edu.tw

Abstract. System-on-a-chip (SoC) has emerged to become a cost-effective approach for embedded systems design with rapid advance of semiconductor technology. It allows designers to integrate a number of heterogeneous IP blocks together based on a system interconnect. However, traditional dedicated wiring as the system interconnect has many shortcomings, such as non-scalable global wire delay, failure to achieve global synchronization, and errors due to signal integrity issues. These problems can be mitigated by the network-on-chip (NoC) architecture based on regular on-chip communication networks. In this paper, we present three efficient switch designs for NoC systems based on circuiting switching. Such switch designs with efficient buffer management can provide the on-chip network with guaranteed throughput and transmission latencies.

1. Introduction

System-on-a-chip (SoC) designs provide designers to integrate a number of IP blocks together. These IP cores can be a processor, DSP, FPGA block, or embedded memory. With rapid advance of semiconductor technology, the complexity of such SoC increases as a result. By the end of this decade, a single chip will accommodate up to 1-billion transistors. Thus, the number of IPs in SoC designs can scale from a few dozens to several hundreds or even thousands [1], [5].

One of the challenges in the billion-transistor era is the communication infrastructure between heterogeneous cores having different characteristics. The interconnect in current SoC based embedded systems for connecting IPs is typically dedicated wires or shared buses. The dedicated wiring approach provides the best communication performance, but its design has poor reusability and scalability. Furthermore, the wire latency and noise significantly affects the reliability of systems when the system complexity increases and the feature size decreases. The shared bus architecture provides a pool of bandwidth among all the cores in the system. However, the shared bus limits the growth of system complexity. To deal with the interconnect problems in SoC chips, a new methodology has to be developed for the next-generation SoC paradigm [2], [3], [4], [5].

To overcome the communication problems of SoC designs, network-on-chip (NoC) with regular tile-based architectures has been proposed for interconnecting the IPs in SoC. Such on-chip interconnection networks provide a high-performance chip-level communication infrastructure with regularity and modularity. Fig. 1 shows the communication infrastructure of NoC with a 2D-mesh tile-based architecture [2], [3], [4], [5].

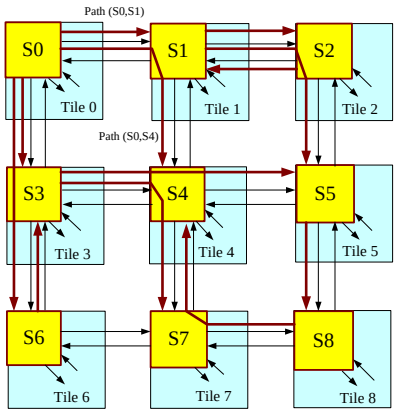

Fig. 1. The communication infrastructure of NoC with a 2D-mesh tile-based architecture. The dotted lines indicate some of the circuit paths provided by the circuit-switched network

The major component in the communication infrastructure of NoC is the routing switch. In Fig. 1, each switch connects up to four neighboring switches and an associated local tile. The switch forwards packets from an input port to the destined output port according to the routing result of the network. In many applications such as multimedia, mobile component, telecommunication and consumer electronics, the characteristics of the communication traffic between IPs can be diversified. The switch needs to efficiently utilize the limited bandwidth of the links to deliver communication data between IPs and satisfies the different requirements of IPs. However, VLSI design is very cost-sensitive, and the required communication quality-of-service (QoS) must be achieved at a reasonable cost.

The circuit-switched network is suitable for NoC architectures since the transmission latency and throughput can be guaranteed. Furthermore, the complexity of the switch architecture in the circuit-switched network is low since its main function is to connect the incoming links to the outgoing links. The drawback of circuit-switched networks is locking resources for the duration of the data transfer. However, this can be alleviated by using virtual channel networks.

In this paper, we address a pre-scheduled circuit-switched network. Such network is suitable for the on-chip network. We also design and implement three efficient switches for such on-chip network. Compared with the design proposed in [9], our first design provides a lower cost. Furthermore, for decreasing the cost and latency, we design other two switch architectures. Our design provides high performance at a reasonable cost.

The next section describes the architecture of the circuit-switched network briefly. In Section 3, we present the design of our switches. Section 4 shows the implementation results. Finally, concluding remarks are given in Section 5.

2. Circuit-Switched Networks for SoC

When a sender wants to deliver data to a receiver in a circuit-switched network, the communication path between the sender and the receiver is established firstly. If there are lots of communication paths in the network at the same time, a link may be shared by multiple paths. Fig. 1 has shown such a network for example. In this diagram, the dotted lines represent the communication paths. The two paths *path(S0,S1)* and *path(S0,S4)* share the link between *Tile0* and *Tile1*. Time-division multiplexing (TDM) technique can solve the problems of sharing links. In TDM, a frame is divided into a fixed number of time slots. When a communication circuit is established across a TDM link, the network allocates a certain number of time slots in each frame of this circuit for the sole use of data transmission.

Before transmitting data, traditional computer networks use the routing probe to establish communication circuits. The routing probe progresses toward the destination to reserve a certain number of time slots for a communication circuit as it is transmitted through intermediate switches. When the probe reaches the destination, the communication circuit has been set up and an acknowledgement is transmitted back to the source. The circuit is released as the data has been transmitted. Fig. 2 shows the time-space diagram of a circuit-switched message. In such architecture, switches need to allocate the valid time slots for the circuit setup. If the network is congested, the setup time t_{setup} might be unacceptable. The traditional computer networks use this approach to achieve guaranteed service, since the communication paths in the cyberspace are too diverse to predict.

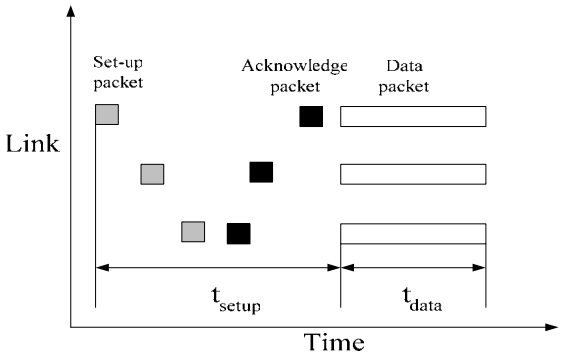

Fig. 2. The time-space diagram of a circuit-switched message

In NoC design, the requirements of communication between IPs are estimated before they are assigned to the tiles. The pre-established circuit-switched network is appropriate for NoC. In many methodologies of NoC approaches [4], [6], [7], [10], a

mapping algorithm is needed to map IPs onto the tiles in the NoC architecture. The mapping algorithm must consider the communication delays and bandwidths between the tiles while it maps the IPs onto the tiles. Thus, after all IPs have been mapped, the communication paths among the tiles of the NoC are potentially existent. Switches can use a table in the switch to store the routing information. Therefore, before transmitting data, switches do not have to set up a communication circuit.

There have been several recent studies on NoC architectures [2], [3], [4], [5], [8], [9], [11], [12], [13], [14], [15], [16], [17]. In [12], the proposed architecture differs from others in that they use the Butterfly Fat Tree architecture as the backbone of NoC. They also design a switch with wormhole routing for their NoC architecture. In [9], a guaranteed-throughput switch is presented. Their switch supports unicast and multicast with pre-scheduling. The input buffers and output buffers in this switch use several frame buffers implemented with random access memory. The cost of their design is too high to be useful in the on-chip network.

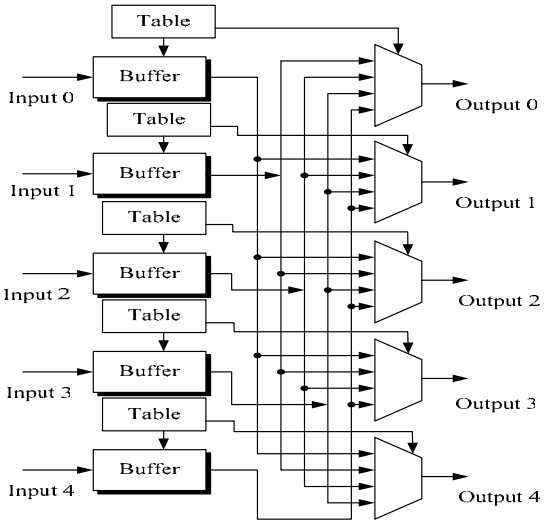

Fig. 3. The diagram of input buffering architecture

Switches connect to up to five links, as *S4* in Fig. 1. An arriving flit may be stored in the buffer firstly. The output port of the switch selects a flit from the buffer and injects it into the network according to the routing table. In the buffering strategy, we distinguish *output buffering* and *input buffering* by the location of buffers inside the switch [13]. An input buffering architecture is shown in Fig. 3. In this architecture, the buffers are at the inputs of the switch. This architecture needs a routing table to determine at which times which buffers are connected to which output ports with contention freedom. In this switch, flits from the input ports are buffered awaiting transmission by the output ports. The size of a frame buffer, FB, in the switch is:

$$FB = BW \times N \times S \quad (1)$$

where *BW* is the link bandwidth, *N* is the number of time slots, and *S* is the interval of the time slot. With a link bandwidth of 1Gbps, the number of time slots of 16, and the interval of the time slot of 2μs, the size of the FB is more than 33Kb. The sizes of the buffer in previously proposed architectures are over than two frames [9], [13].

Since NoC architecture design is cost-sensitive, the buffer size must be decreased as much as possible. We hence propose the switch architecture with only a frame buffer at every input.

We show the diagram of this architecture with one input and its associated output port in Fig. 4. We omit the global signals such as *clocks*, *reset* and some stimulating signals in this diagram for simplicity. In this architecture, each input has a frame buffer (FB) and two *input slot address tables*, ISATa and ISATb. The FB is used to store the input frame data, while the ISATs are used to store the addresses of the input data in the FBs. Each output port has a private *input time slot table* (ITST), and the ITST has two fields, *input port* (IP), *input time slot* (ITS). Furthermore, each input port has a private *output port table* (OPT). The OPT and the ITST determine at which times which buffers are connected to their output port respectively.

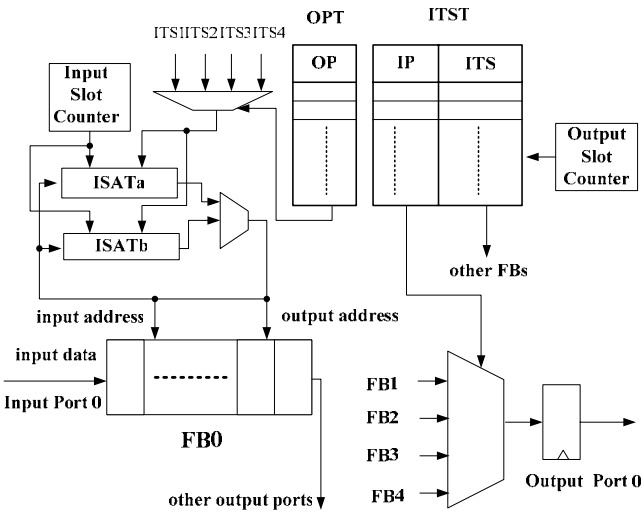

Fig. 4. The switch architecture with a frame buffer at each input

Latency is defined as the duration of a packet being transported from the sender to the receiver. The major factors of the latency are the latency of the switch and latency of the link, denoted by L_s and L_l. The latency for sending one bit of data from tile $tile_i$ to tile $tile_j$ can be analytically calculated as:

$$L_{bit}^{tile_i, tile_j} = \sum_{i \le n_{hops}} L_{S_i} + \left(n_{hops} - 1\right) \times L_l \qquad (2)$$

where n_{hops} is the number of switches the bit passes on its way from tile $tile_i$ to tile $tile_j$. If the latencies of all switches in NoC are equal, the equation (5) becomes:

$$L_{bit}^{tile_i,tile_j} = n_{hops} \times L_s + \left(n_{hops} - 1\right) \times L_l \quad (3)$$

Compared with the architecture in [9], the architecture in Fig. 4 provides a lower cost. However, both switches do not deliver the data of the frame to the next destination until all flits of the frame have already arrived. Thus, the latency of these switches is too long. In NoC architecture, we do not need to do so. For decreasing the latency of the switch, the arriving data has to be transmitted as early as possible.

3. Design of Efficient Switches

In Fig. 5, the architecture is simpler than the architecture shown in Fig. 4. Arriving at an input port, the data is buffered in the temporary register. Without the whole frame has arrived, the data will be stored in the FB or directly sent to output ports. To do so, a comparator is needed to control the direction of the data stored of the temporary register. The comparator compares the input time slot of the data with the output address to generate the control signal which determines the destination of the data of the temporary register. If the data of the temporary register exactly is one of other output ports want to transmit, it is sent to this output port immediately. Otherwise, it will be stored in the FB, since it needs to wait for transmitting with a certain times. The address of the data in the FB is based on the input time slot of the data. The time of the data storing in the FB does not exceed the period of a frame transmission. Thus, a FB is enough to store all input data that need to wait for transmitting.

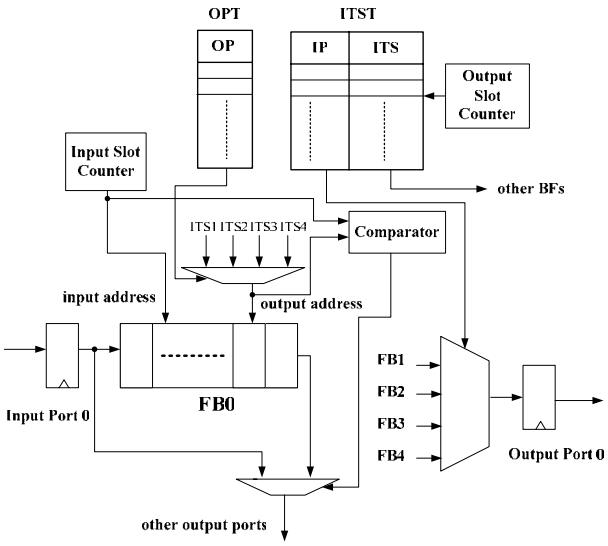

Fig. 5. The switch architecture with one FB at each input

A scheduler is needed to lower the waiting time. By employing optimal scheduling, the latency of data staying in switches can be minimized. Such optimal scheduling is an NP-complete problem. In the design described in the following, it meets the worst case scenario that all input data need to be stored. Therefore, the size of the buffer at each input is one FB.

In this architecture, some locations of the buffer will never be used for storing data, since some data is immediately transmitted. The unused space of the buffers in the switche is wasted. Though the size of the buffer depends on the scheduling algorithm, we also propose a second switch design in this section that reduces the cost of the switch. In this architecture, the buffer size at each input is less than a FB according to the application and can be designated by the designer.

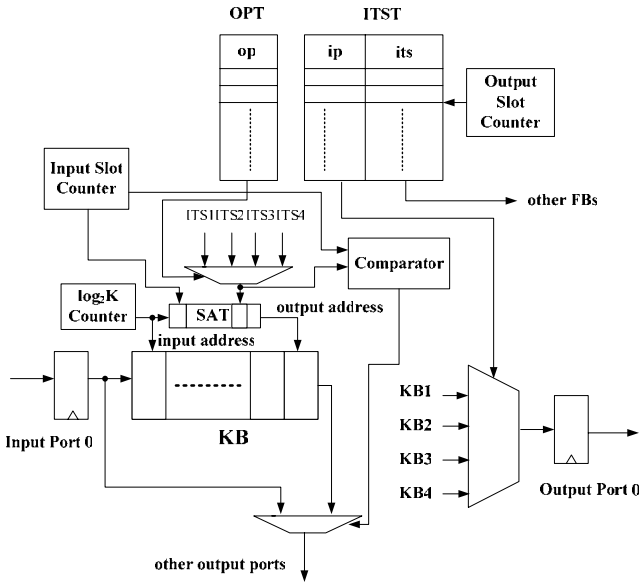

Fig. 6. The switch architecture with less than one FB at each input

Fig. 6 shows the low-latency and cost-effective switch architecture with less than one FB at each input. In this architecture, the size of the input buffer, denoted by KB, is designated by the design according to the specific application. The *slot address table* (SAT) stores the addresses of the data that needs to be buffered in the KB according to the input time slots of the data.

When arriving data needs to be stored into the KB, the $\log_2 K$ Counter provides an address. The size of the counter is at least $\log_2 K$ bits. The KB uses this address to store the data. Meanwhile, SAT stores this address in the location corresponding to the input time slot of the data. In each output time slot period, if one of the data in the KB is to be sent to the next switch, the SAT outputs the address of the data according to the ITS selected by the OP of the OPT. The data of this address in the KB is sent to the scheduled output port.

The size of the SAT is $S \times \lceil \log_2 K \rceil$ bits, where S is the number of time slots in a frame. With S of 32, K of 16, the size of the SAT is 128 bits. Such size is smaller than a location of the buffer in the switch. Hence, the cost of this architecture is significantly reduced.

4. Implementation

We have implemented the designs described in the above. We use pipelining technique to improve the performance. This architecture has three pipeline stages. The number of the time slots in a frame is 16. The data bus of the input port is 128 bits. We need one clock cycle for a packet crossing a link. Thus, the packet size is 128 bits. The table setup circuit is responsible for setting up the OPT and ITST and forwards the setup message to the next switch. Before initializing the system, the data for the setup table must be pre-scheduled and generated by a software scheduler. The scheduler could be part of the EDA tools for the NoC platform. On the other hand, we can also embed the scheduler into the chip and run be a processor core. Such design can dynamically schedule the circuits in NoC. Our design can support both scheduling methods.

Table 1 illustrates the chip area of three designs. Each is a 5 × 5 switch with a clock rate of at least 1 GHz supporting 16 time slots. The link width of the first and the third architectures is 128 bits while the second is 32 bits. The size of the buffer at each input of the switch for the first and the second architectures is a frame while the third is half a frame. In the experiment results, the third architecture reduces the cost significantly.

Table 1. Chip areas of three designs

Switch Architecture	Chip Area
128 bits with one frame buffer at each input	1.48mm^2
32 bits with one frame buffer at each input	0.4mm^2
128 bits with half a frame buffer at each input	0.26mm^2

Table 2. The comparison between our design and others

Design	Link Width	Freq.	A. Bandwidth	Area	Technology
Ours	128	1GHz	640Gb/s	0.26mm^2	0.18μm
[8]	32	185MHz	29.6Gb/s	3.5 mm^2	0.25μm
[13]	32	500MHz	80Gb/s	0.175mm^2	0.13μm

We have used TSMC 0.18μm technology to implement our designs. The layout of the first design in the table is shown in Fig. 7. Our chip has an aggregate bandwidth of 5 × 1 GHz × 128 bits = 640 Gbit/s. Table 2 shows the comparison between our design and others. In [8], a switch for multiprocessor with latency insensitive NoC is proposed. In [13], their switch also provides both circuit-switched and

packet-switched transmissions. To reduce area, they have implemented a dedicated FIFO architecture with full-custom design. However, our design achieves higher throughput with less advanced technology.

5. Conclusions

In this paper, we propose the pre-scheduling circuit-switched network for the on-chip network in embedded system design. Such network does not set up communication paths before transmitting data. Thus, it provides guaranteed transmission throughputs and latencies. We have designed three efficient switch architectures to support this network. In these architectures, we show the trade-offs between hardware complexity and efficiency. We have used TSMC 0.18μm technology to implement our designs. The layout of one of the designs has an area of 0.26 mm^2 and offers 640 Gbit/s aggregate throughputs.

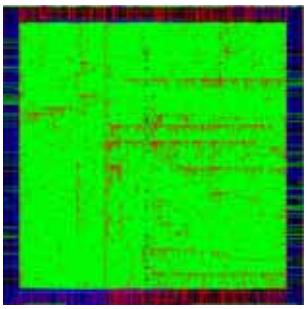

Fig. 7. The layout of the design of 128 bits with one frame buffer at each input

References

1. Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, World Semiconductor Council, 1999.
2. L. Benini, G. De Micheli, “Networks on Chips: A New SoC Paradigm,” *IEEE Computer*, vol. 35, pp. 70 -78, Jan. 2002.
3. W. J. Dally and B. Towels, "Route packets, not wires: On-Chip Interconnection Networks," *Proceedings of 38th Design Automation Conference*, pp 684-689, June 2001.
4. S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, “A Network on Chip Architecture and Design Methodology, ” *Proceedings of the IEEE Computer Society Annual Symposium on VLSI, 2002 (ISVLSI.02)* , pp 105-112, April 2002.
5. A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist, “Network on Chip: An Architecture for Billion Transistor Era,” *Proceedings of IEEE NorChip Conference*, pp. 166-173, Nov. 2000.

6. J. Hu and R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures," *Proceedings of Design Automation and Test in Europe Conference and Exhibition*, pp. 688-693, March 2003 .
7. T. Lei and S. Kumar, "A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture," *Proceedings of Euromicro Symposium on Digital System Design: Architectures, Methods and Tools, Turkey*, Sept. 2003.
8. L.-Y. Lin, C.-Y. Wang, P.-J. Huang, C.-C. Chou, J.-Y. Jou, "Communication-driven Task Binding for Multiprocessor with Latency Insensitive Network-on-Chip," *Proceedings of Asia and South Pacific. Design Automation Conference 2005*.
9. J. Liu, L.-R. Zheng and H. Tenhunen, "A Guaranteed-Throughput Switch for Network-on-Chip," *Proceedings of International Symposium on System-on-Chip*, Nov. 2003.
10. S. Murali and G. D. Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures," *Proceedings of Design Automation and Test in Europe Conference and Exhibition*, pp. 869-901, Feb. 2004.
11. J. Nurmi, I. Saastamoinen, and D. Siguenza-Tortosa, "Interconnect IP Node for Future System-on-Chip Designs," *Proceedings of 1st IEEE International Workshop on Electronic Design, Test and Applications (DELTA '02)*, Jan. 2002.
12. P. P. Pande, C. Grecu, A. Ivanov, R. Saleh,"Design of a Switch for Network on Chip Applications," *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, Jan. 2003.
13. E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. Van Meerbergen, P. Wielage, and E. Waterlander, "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip," *Proceedings of Design Automation and Test in Europe Conference and Exhibition*, March 2003.
14. M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, and J. Rabaey, and A. Sangiovanni-Vencentelli, "Addressing the System-on-a-Chip Interconnect Woes through Communication-Based Design," *Proceedings of 38th Conference on Design automation*, June 2001.
15. S. Sathe, D. Wiklund, D. Liu, "Design of a Switching Node (Router) for On-chip Networks," *Proceedings of the 6th International Conference on ASIC (ASICON 2003)*, Oct. 2003.
16. D. Wiklund and D. Liu, "SoCBUS: Switched Network on Chip for Hard Real Time Embedded Systems," *Proceedings of International Parallel and Distributed Processing Symposium*, April 2003.
17. D. Wingard, "MicroNetwork-Based Integration for SOCs," *Proceedings of 38th Design Automation Conference*, June 2001.