Elastic Flow Control and Parallel Switch Design for SpaceWire Router

Session: SpaceWire Networks and Protocols (Poster)

Long Paper

Chunjing MAO, Yong GUAN

College of Information Engineering, Capital Normal University, Beijing, 100048, China/ Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China

Zili SHAO

Department of Computing, The Hong Kong Polytechnic University,

Hung Hom, Kowloon, Hong Kong, China

Jie Zhang

College of Information Science & Technology, Beijing University of Chemical Technology, Beijing, 100029, China

E-mail: mcjing@163.net, gxy169@sina.com, cszlshao@comp.polyu.edu.hk, jzhang@mail.buct.edu.cn

ABSTRACT

In a SpaceWire network, to connect equipments together, a SpaceWire router uses the wormhole routing to deliver packets. However, in the wormhole routing, there inherently exists the braking-problem that increases the average non-blocking latency. In this paper, we propose an elastic flow control mechanism to solve this problem. In addition, we propose a novel parallel switch architecture with pipelining to improve the transmission speed and switching efficiency, and optimize its FPGAs implementation. We implement our technique and test it in Xilinx FPGAs. The results show that on average the non-blocking latency can be reduced to 245ns at 200MHz. By using our elastic flow control, a long physical connection between two nodes can be established without reducing the bandwidth utilization. With our pipelined parallel switch architecture, the transmission speed can be enhanced to over 300Mbps and the circuit scale can be reduced 50% compared with the original design in FPGAs implementation.

1 WORMHOLE ROUTING

Wormhole routing is an effective solution for packet routing [1][2]. Each packet contains a header which holds the destination node address. As soon as the header for a packet is received, the router determines the output port to route the packet to by checking the destination address. If the requested output port is free, the packet is routed immediately to that output port. That output port is now marked as busy until the last character of the packet has passed through the router – indicated by the end of packet marker being detected by the router. Wormhole routing only cuts down on the amount of buffering used within each router and the delay for packets deliver.

Compared to a store and forward technique, where an entire packet is first received and stored before it is sent out of the router, NoC (Network on Chip) [32] can be realized. And delay of a single packet T_{lower} can be described as:

$$T_{lower} = (L_f / B_{ch}) \cdot D_{sd} + L_p / B_{ch} \qquad (1)$$

L_f is the length of each flit, D_{sd} is the distance between source node and destination node, B_{ch} is the bandwidth of the channel, and L_p is the length of packet. If $L_f << L_p$, the influence from D_{sd} to T_{lower} can be ignored.

Wormhole routing [1] is illustrated in Figure 1 which shows a packet being sent from one node to another through a routing switch (router). The header of the packet is marked as black, while the rest of the packet is marked as grey. As soon as the router receives the header, it checks the requested output port. If the output port is free, then the router makes a connection between the input port and the output port. The packet then flows through the router. When the end of packet (EOP or EEP) marker is received by the switch, the router terminates the connection and frees the output port for the next packet, which can come from any input ports.

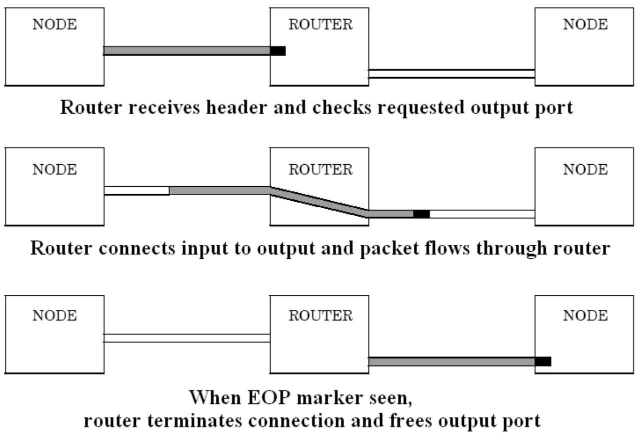
NODE
ROUTER
NODE
Router receives header and checks requested output port
NODE
ROUTER
NODE
Router connects input to output and packet flows through router
NODE
ROUTER
NODE
When EOP marker seen,
router terminates connection and frees output port

Figure 1. Wormhole Routing

Using of blocking flow control mechanism in wormhole routing, there is an inherent braking-problem [4][5]. When a header flip reaches the router, the flip has to wait until next channel is free. Transmission time of packet from header flip to end flip is T_b, and it can be expressed as:

$$T_b = T_{AD} + T_{ARB} + M \times T_s \qquad (2)$$

T_{AD} is coding time of address coder; T_{ARB} is arbiter time, and T_S is blocking signal transmission time in router. Because all the data flips are transmitted with pipelining mode, clocking time T_c need to be greater than T_b (see Equation 3). Otherwise, packet-loss would happen.

$$T_c > T_{AD} + T_{ARB} + M \times T_s \qquad (3)$$

Latency built up in router is with millisecond. A signal transmission can be finished with nanosecond, and clocking time is in millisecond. From Equation 3 we can see that, working-frequency is limited by the braking-problem, and working-frequency is one of the key factors to improve the transmission rate.

2 Elastic Flow Control

Based on analysis above, an elastic buffer is applied to solve the brake problem in wormhole routing. Elastic flow control uses flip as the basic unit. In this paper, we propose an elastic flow control mechanism to illustrate the idea of design elastic buffer.

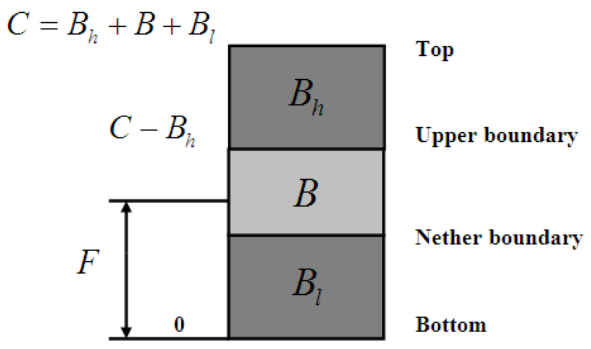

Figure 2. Structure of Elastic Buffer

As is shown in Figure 2, the capacity of buffer is C, the upper boundary is B_h, and the lower boundary is B_l. F is the number of bytes stored in a buffer, and $F = 0$ initially. While outgoing transmission rate is equal to incoming transmission rate, F maintains a fixed value. While outgoing transmission rate is lower than incoming transmission rate, the value of F will gradually increase. When it reaches the upper boundary, a flow control signal is built to notify the incoming node to stop transmission, and here remain B_h unused bytes in buffer for storing the incoming data in braking time. The braking distances are determined by B_h. By increasing B_h, braking distances can be prolonged.

Accordingly, the setting of B_l is to prevent flow breaking while read-out from buffer. While the value of F reaches nether boundary, flow control signal can be revoked to notify incoming node continuing transmission. It needs time for low control signal to be revoked from incoming node, and with B_l, it can prevent flow breaking when low control signal be revoked. Thus transmission would be more effective.

By using elastic flow control, address coder time T_{AD} and arbiter time T_{ARB} do not need to be calculated, and delay can be reduced by paralleling data transmission and channel switch. The flow control signal is built and revoked at the same time, which is equal to blocking time T_B. So T_c should be:

$$T_c > T_B + T_s \qquad (4)$$

It is easy to build a flow control signal, which can be finished within several nanoseconds. Compare with Equation 3 and Equation 4, braking-problem can be solved with elastic flow control. Also, there can be a long physical connection between two nodes without reducing the bandwidth utilization by B_h and B_l in synchronous mode.

3 Parallel Switch Architecture

Normally, there are three steps for a packet passing from input port to output port in SpaceWire router:

A. Reading header for the packet and sending the destination node address to routing table

B. Finding destination node address in routing table and to decide the output port of the packet

C. Sending the packet to the decided output port

There is a read/write competition when more than two input ports sending packet to the same output port. This would trigger the arbitral mechanism in router. After arbitration, the input port with higher priority can send its packet. That is, only one input port can send packet at a certain time; others would sending packet sequentially by arbitral result. Transmission efficiency of router would be confined by this serial arbitration mode.

We design a pipeline based SpaceWire router to improve the transmission efficiency. The objective is to design a pipeline based non-blocking parallel switch, as is shown in Figure 3.

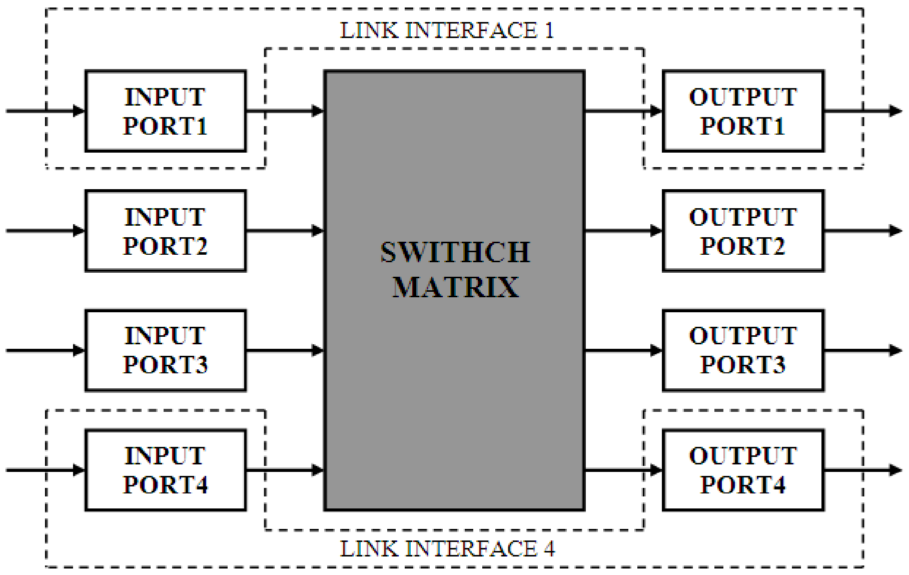

Figure 3. Illustration of Switch Matrix

Circuit of packet header detection and packet reorganizing can be very complex with a large circuit scale in original SpaceWire router design. We propose a transaction processing pipeline for packet header detection and packet reorganizing, which can reduce the complexity of circuit design and power consumption, and improve the reliability of SpaceWire router. The architecture of parallel switch pipelining is illustrated in Figure 4. We realize this architecture with FPGAs, and experiment results show that, by using transaction processing pipeline, the circuit scale can be reduced about 50% compare to original design.

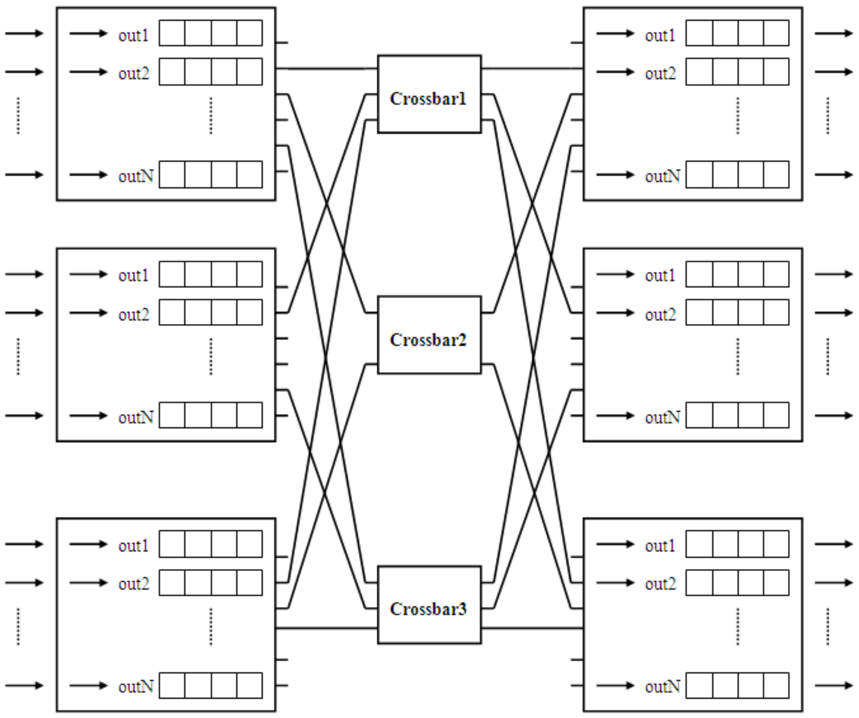

Figure 4. Architecture of Parallel Switch Pipelining

As illustrated in Figure 4, this architecture composes of K no-buffer crossbar switches and M input sharing storage modules. There are V external interfaces for each input/output sharing storage module, which connects to all crossbar switches.

There are $M \times V$ VOQs (Virtual Output Queue) in each input sharing storage module, which stores packets to different destination. V output queues in each output sharing storage module store packets wait be sent. In each arbitration cycle, input port picks up M queues from $M \times V$ VOQs randomly, and submits scheduling requests to switch. Switch services each queue by polling and feedbacks authority information to input port by scheduling result. The first packet of VOQs that appointed by authority information will pass switch to output queues, and it will be sent to external links after message reorganization. Push-reverse mechanism can be adopted to avoid overflow in output.

In case of multi-input sending packets to one output, we adopt pipeline technology for multi-transmission by time-sharing operation. Also, routing table can copy to each input port, and there is no need for arbitral mechanism. While a packet arrives in input port, it decides the output port by finding in routing table by header of packet, and it will be sent the packet. While multi-input transmission, it would assign a time-token for each input port. Packet can be transmitted when time-token of this input port is enabled, and transmission is stopped once time-token is disabled. All incoming packets can follow this way. Assigning time-token can be controlled by routing algorithm. The basic idea is to assign time-token to each input port alternately by using pipeline, and routing algorithm can be optimized in some specific application. It would improve transmission performance in SpaceWire network, and accessing for each input/output port can be achieved simultaneously.

4 Experiments

We conduct experiments by building a platform with Xilinx FPGAs, which is shown in Figure 5. The platform consists of two routers, and each router is connected with several nodes. Each node includes the data source of video, image, audio, and instructions. Each router connects with a PCI node, which is used to connect the host PC. The host PC can observe and configure the router via PCI node.

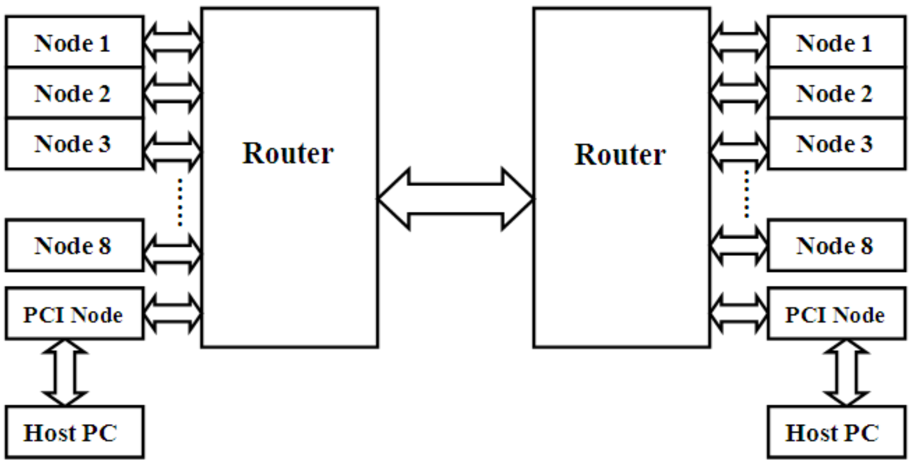

Figure 5. Illustration of Experiment Platform

The experimental results show that on average the non-blocking latency can be reduced to 245ns at 200MHz. By using our elastic flow control, a long physical connection between two nodes can be established without reducing the bandwidth utilization. With our pipelined parallel switch architecture, the transmission speed can be enhanced to over 300Mbps and the circuit scale can be reduced 50% compared with the original design in FPGAs implementation.

5 Conclusion

In this paper, we solve the braking-problem by using elastic flow control mechanism. We proposed novel parallel switch architecture with pipelining to improve the transmission speed and switching efficiency, and we optimize its FPGAs implementation. The experimental results show that on average the non-blocking latency can be reduced to 245ns at 200MHz, and transmission speed can be enhanced to over 300Mbps. The circuit scale can be reduced 50% compared with the original design in FPGAs implementation.

References

1. "SpaceWire protocols", ECSS-E-ST-50-11C, July 2008

2. "SpaceWire standard - Links, nodes, routers and networks", ECSS-E-ST-50-12C, 31 July 2008

3. Ai Zhen. “Design of Wormhole Routing Based on Black-bus,” Master Degree Thesis of University of Electronic Science and Technology of China, 2007(5).

4. An Xuejun, Zhu Faming, Gao Wenxue, Wu Dongdong. “The Design and Implementation of the Elastic Buffer in Wormhole Routing Chips,” Computer Engineering and Applications, 2002, 7.

5. Xiao Xiaoqiang, Jiang Yuqin, Jin Shiyao, He Hongjun. “BWR—Buffered Wormhole Routing Switching,” Chinese Journal of Computer, 2001, 24(1).