International Journal of Computer & Information Technologies (IJOCIT)
www.ijocit.ir & www.ijocit.org
ISSN = 2345-3877

Phoenix-S2:
A Low Latency Switch for NoC based on FSM and Locking Mechanism

Akram Reza[1], Midia Reshadi[2]

Sama technical and vocational training college[1], Department of Computer Engineering[2]
Islamic Azad University[1,2]
Karaj Branch[1], Science and Research Branch[2]
Karaj[1], Tehran[2]
Iran[1,2]

(Email: ak.reza@hotmail.com)

Abstract: Disproportion between gate and wire delays on chip is aggravated by technology shrinking in the deep submicron (DSM) domain. Thus, Network on Chip is proposed to address DSM problem. In this paper we propose a low latency NoC switch based on Deterministic Finite State Machine and port locking mechanism which is implemented by multiplexer. Routing and arbitration operations are done in only one clock. In addition, switch area is reduced significantly in comparison to conventional NoC switches. Proposed Switch is described in VHDL; validation and simulation are performed by ModelSim. Synthesis is executed by Leonardo Spectrum-98 tool in ASIC and FPGA designs.

Keywords: Network on chip; switch; Finite State Machine; buffering strategy.

1. Introduction

Network on Chip (NoC) is proposed as a new design methodology [1-3]; to replace traditional bus architectures and to provide higher bandwidth, lower latency and scalable communication infrastructure for modernsystems on chip. NoC design imposes several challenges. Elements of NoC must be fast; due to this routing, switching, and links have to be simple and efficient. On the other hand, area and power limitations, scalability, reusability, and reliability are, also, serious [4-6]. Switching and routing techniques absolutely affect the switch structure. So, it is an important constituent of NoC architecture which has a straight influence on many design parameters such as end to end delay, packet latency, power consumption, and peak performance [6 and 7].

From the other point of view, switches as a main element of NoC consume around 58% of whole NoC area [6]. In addition, practical results indicate that buffers in switch dominate switch area. Another parameter of switch design is a latency, which affects on-chip network performance. More or less in NoC switch, this parameter is more restricted to arbitration and routing logic depended on their policies.

In this paper, we present structural switch architecture which is suitable for NoC architecture. A key aspect of proposed design is applying Finite Sate Machine (FSM) [8]. Inasmuch as, FSM implementation of different parts of switch makes our design compact and fast [9]. The switch was described in VHDL. Cycle-accurate simulation appropriately presents state transitions, ports pipelining functionality, and processing of port locking. Phoenix-s2 switch design is general and flexible in the terms of using in various topology and engaging different routing and arbitration policies. We evaluate our design in area consumption and delay standpoints. Results indicate that arbitration and routing latency span only one clock. Hence, we achieve low latency switch design. Eventually we examine proposed switch area with different port numbers and buffering strategies.

This paper is organized as follows. First, in section 2 we introduce Phoenix-s2 switch architecture and related Finite State Machines. Chip area and buffer analysis and synthesis results are presented, in the section 3. Finally, we conclude our paper in section 4.

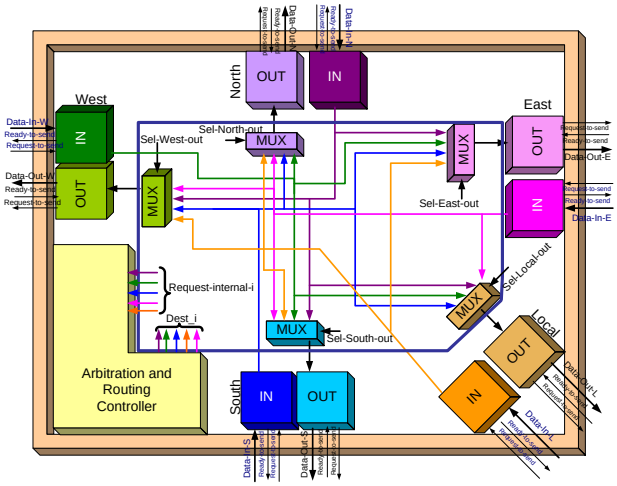

Figure 1: Phoenix-s2 switch structure

2. Switch Architecture

In the current paper we consider mesh topology and packet switched network, but the prototype switch can be used for any topology which needs a five ports switch. Also, switch can be deployed for different port numbers. As, we deploy switch with three and four ports. Prototype switch has five ports as N (North), W (west), S (south), E (east), and L (local), (See Figure 1).

Local port connects to core. The switch under test implements deterministic routing and wormhole switching. However, any routing algorithms can be utilized. Due to the fact that, routing logic is absolutely independent from other parts of switch architecture, especially arbitration logic and input ports, (See Table (1)).

Although, flits of transmitted packets are firmly one byte, due to wormhole switching; Numbers of data flits in every packet are variable, due to achieve high compatibility to most NoC topologies, applications, and switching techniques through network. Therefore, packet is composed of three fields which are shown in Figure 2:

- First flit (8 bits): Destination Address
- Second flit (8 bits): Numbers of data Flits
- Payloads: Flits of Data

Table 1: Arbitration pseudo code

```
IF request_internal_north='1' THEN
        dest_north is determined by routing algorithm
        request_internal_north<='0';
ELSIF request_internal_west='1' THEN
        dest_west is determined by routing algorithm
        request_internal_west<='0';
ELSIF request_internal_south='1' THEN
        dest_south is determined by routing algorithm
        request_internal_south<='0';
ELSIF request_internal_east='1' THEN
        dest_east is determined by routing algorithm
        request_internal_east<='0';
ELSIF request_internal_global='1' THEN
        dest_gloabl is determined by routing algorithm
        request_internal_global<='0';
END IF;
```

We employ both input buffering, and, also, input/output buffering strategy. Finally selection between these two strategies is completely dependent on specified applications which are transmitted via proposed switches. Synthesis results will be discussed in the terms of both area and timing in section 3. Hence, in input buffering strategy, every port has two buffers. The first one receives data from *data_in* signal and the second one forwards data to *data_out* signal. Each buffer has eight columns which are adapted to flits size. Although, studies indicate that six rows are proper for most applications [6]; Purposely, in presented switch, row numbers are three. This is chosen due to the pipeline operation among

three levels of transmission, low latency arbitration process, and buffer area consumption which is very critical. Although simulation shows us more than three rows are not optimized, number of buffer rows can be selected directly by considering specified applications constraints and topology limitations.

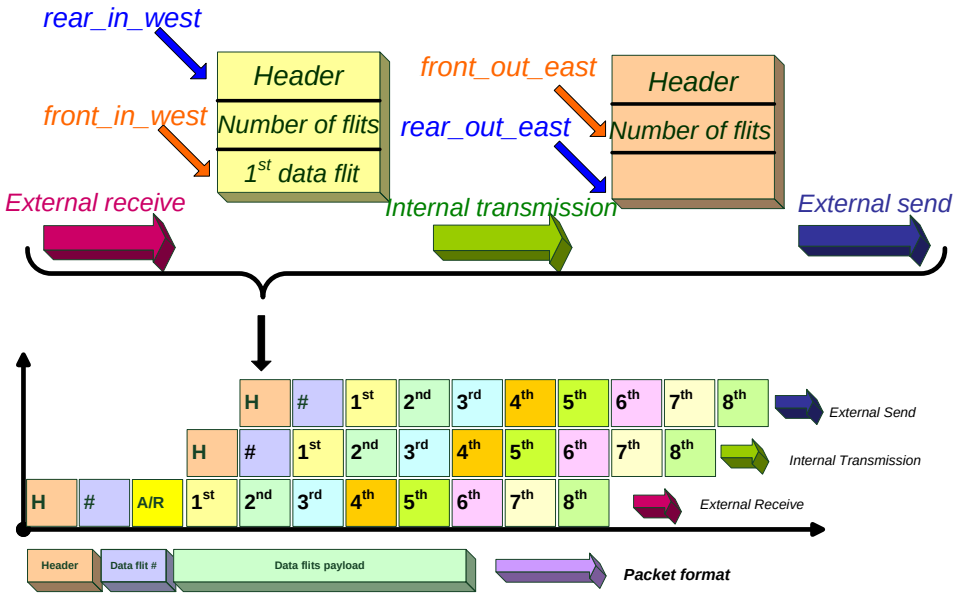

Figure 2: Packet forwarding via pipeline process in the switch is shown

Each flit (8 phit) in a packet is stored just in one clock. Fundamentally, we do not consider link latency in this design level, due to hierarchical design. Figure 2 presents the data transmission through the NoC switch and also outside it. Correspondingly, packets, flit by flit, are fed into input buffer via *data_in* signal. Then, they are dispatched out to match output port, based on arbitration process, routing process, locking, and internal transmission; eventually, they are sent out to *data_out* signal. Consequently, three transmission stages are required. First of all, external received from previous switch or core, secondary, internal transmission in switch, and finally external sending to next switch or core.

3. Finite State Machine in switch structure

The data transmission is controlled by two finite state machines: Receive, and Send. These machines work concurrently with each other per clock to perform punctual operation, to construct switch more structural, and to increase visibility of the switch behavior in the peak work load. Remember, when all machines are in init state, it shows ports preparation. Hence we do not need to check lock signals to enter current FSM.

Receive machine in port (i) is commenced by an external request from neighbor switch which indicates neighbor switch has a packet for transmission (see Figure 3).

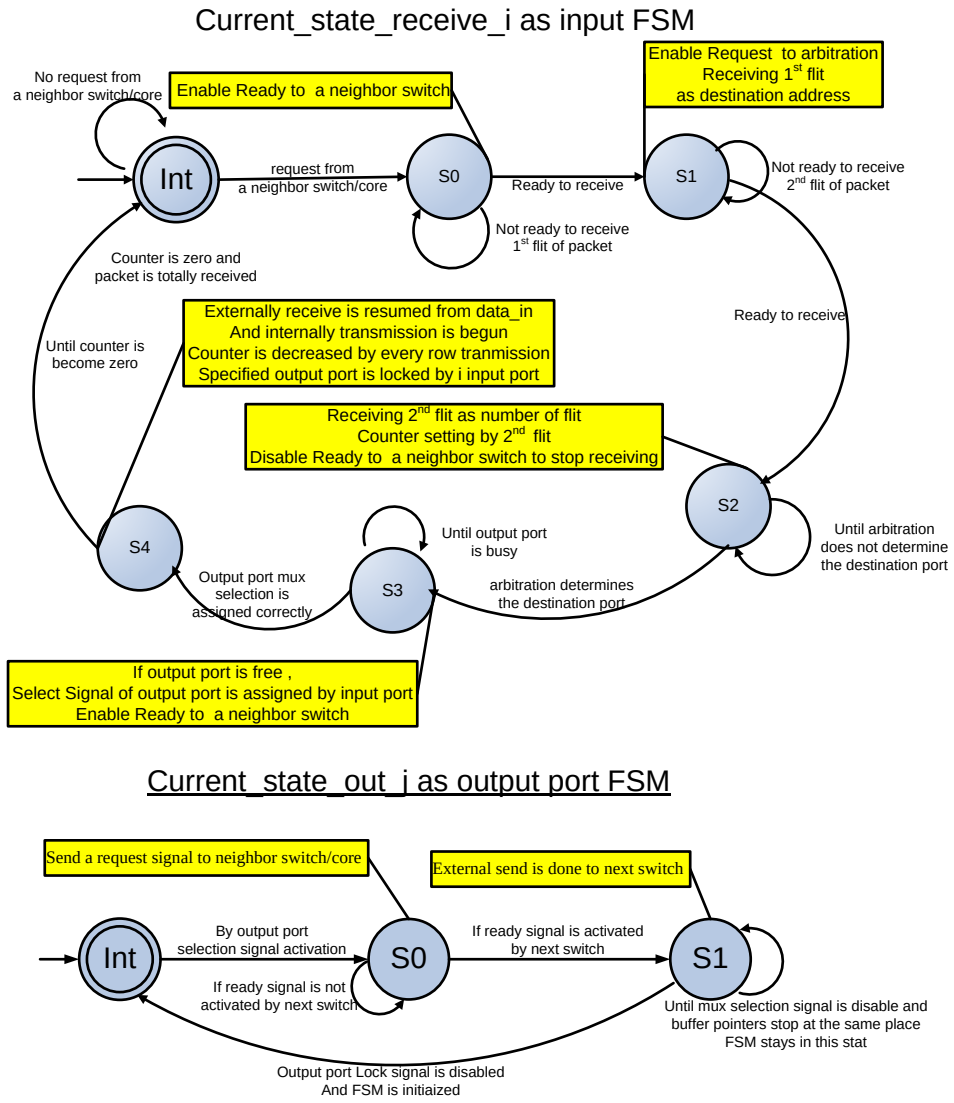

Figure 3: Receive Finite State Machine for (i) input port and Send Finite State Machine for (j) output port.

Therefore, receive machine is triggered to S_0 and replies to neighbor switch its preparation. Then it is triggered to S_1 sending an internal request to arbitration and receiving first flit of the packet as destination address. Precisely, just at this time, arbitration realizes which input port is active and by its destination address, it can determine the proper output port. The receive machine in port (i) goes to S_2 to receive second flit as a number

of data flits, and, also, counter is initialized and ready signal is disabled to cease external receive from neighbor switch for a clock. At S_3, arbitration determines the output port (j) just in one clock and sends it to "receive" machine. In this state if output is free, depending on input, output multiplexer "select" signal is assigned. Therefore, output is locked. Also, ready signal to neighbor switch is enabled to show that input (i) is ready and specified output is decided.Then input machine is triggered to S_4. Hence, external receive from neighbor switch is resumed and internal transmission between in and out buffers is commenced. Counter in every input is decreased by every phit transmission. Therefore, machine stays in **S4** until input counter reaches to zero.

Simultaneously, send machine in output port is commenced by initializing multiplexer select signal with proper input port number, (See Figure 3). Hence, output (j), in **S0,** sends a request to next neighbor switch; as a result, external handshake flow control is begun. By receiving its ready, in S_1 state, external send is commenced and it is continued until selection signal is initialized by input port which means the termination of process.

4. Experimental Results

The Phoenix-s_2 NoC switch is described in VHDL. ModelSim simulation tool is used to validate proposed switch functionality Moreover, the switch area consumption was estimated by varying two parameters: number of port and buffering strategy. The Leonardo Spectrum-98 tool was used to synthesize the Phoenix switch in two different technologies: Xilinx Virtex FPGAs and XCLo5U ASIC. Synthesis was conducted with maximum area and delay. We deployed three rows in every buffer, because in simulation we realized that more than three is not optimized in our switch structure. As you see in both table 2 and table 3, we synthesized proposed switch with (in) and (in/out) buffering strategies. Since, proposed switch can be utilized in any topology that may have different applications, these two

strategies can be used depend on specified applications. Phoenix-s2 in comparison to conventional switches is more optimized in area and delay.

Table 2: Phoenix switch synthesis results for (In/Out buffering) strategy for different port numbers

Parameters \ Number of Ports	3 ports	4 ports	5 ports
ASIC clk with area optimization	133.1MHz	103.6 MHz	24.4 MHz
ASIC gate number with area optimization	6384.0	10590.3	17020.9
ASIC clk with delay optimization	181.5 MHz	147.6 MHz	63.3 MHz
ASIC gate number with delay optimization	8612.3	14386.0	26168.8
FPGA clk with area optimization	22.8 MHz	16.4 MHz	4.4 MHz
FPGA LUT with area optimization	531.0	1033	1630.0
FPGA clk with delay optimization	29.6 MHz	20.5 MHz	6.0 MHz
FPGA LUT with delay optimization	671.0	1150	1698

Table 3: Phoenix switch synthesis results for (In buffering) strategy for different port numbers

Parameters \ Number of Ports	3 ports	4 ports	5 ports
ASIC clk with area optimization	177.3 MHz	101.6 MHz	29.5MHz
ASIC gate number with area optimization	2389.6	6503.0	9439.1
ASIC clk with delay optimization	277.8 MHz	147.0 MHz	65.4 MHz
ASIC gate number with delay optimization	3686.8	11206.4	17611.7
FPGA clk with area optimization	42.6 MHz	14.5 MHz	4.4 MHz
FPGA LUT with area optimization	152	659.0	1032.0
FPGA clk with delay optimization	44.5 MHz	21.5 MHz	6.6 MHz
FPGA LUT with delay optimization	162.0	789	1264.0

5. Conclusion

In this paper we present a low latency, parameterizable, and structural switch architecture suitable for on-chip networks, based on Finite state machine. Phoenix-s2 switch can be used for any topology which needs five ports switch. Also configuration concept of prototype switch can be deployed for different port numbers and various routing and arbitration policies depends on application and topology of on chip network architecture. In order to achieve a structural design, we use finite state machine concept. The prototype switch is presented for mesh topology with wormhole switching mechanism. The evaluation results show that the routing and arbitration are performed only in one clock. We describe the structure of switch based on two FSM: receive and send. In addition, proposed switch is described in VHDL and simulated with ModelSim to validate switch functionality. We also synthesize switch in the term of number of port and buffering strategies. Results show that, proposed switch is efficient in

area and delay in comparison to conventional NoC switches. For future works we are planning to analyzing different arbitration mechanisms in switch.

References

[1] L. Benini, G. De Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] W. J. Dally, B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," Proc. 38th Design Automation Conference (DAC), pp. 684- 689, June, 2001.

[3] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, D. Lindqvist, "Network on a chip: an architecture for billion transistor era", Proc. IEEE NorChip Conference, November, 2000.

[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," Proc. ISVLSI'02, Pittsburgh, USA, pp. 105-112, April, 2002.

[5] L. Benini, G. De Micheli, *Networks on Chips: Technology and Tools*, Morgan Kaufmann, San Francisco, CA, 2006.

[6] F. Moraes, N. Calazans, A. Mello, L. MÖller, L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *INTEGRATION, the VLSI journal*, Vol. 38, pp. 69–93, 2004.

[7] M. Millberg, E. Nilsson, R. Thid, S. Kumar, A. Jantsch, "The Nostrum backbone - a communication protocol stack for networks on chip," Proc VLSI Design Conference, Mumbai, India, January, 2004.

[8] J. Hopcroft, J. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, june, 2001.

[9] J. Duato, S. Yalamanchili, L. Ni, *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann, Los Altos, CA, 2002.

[10] Model Technology, ModelSim Foreign Language Interface, Version 5.5e, 2001.

