IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Design Round Robin and Interleaving Arbitration algorithm for NOC

Boppidi Srikanth
Srikanth.vlsi.2011@gmail.com

Abstract

NOCs (Network on Chip) are increasing in popularity because of their advantages: larger bandwidth, and lower power dissipation through shorter wire segments. Communications in large SOCs are so important that many designers have adopted the NOC approach. The challenges consist in offering the best connectivity and throughput with the simplest and cheapest architecture methodology.

In this project it is proposed to Design multi-module architecture (Masters or Slaves) for Network On chip configuration. When multi-modules are involved, there is a need to solve the problem of bus contention among the modules, masters and slaves. To resolve this bus contention problem, two algorithms, interleaving and normal Round robin algorithms in this architecture. In the normal mode, the Round Robin design architecture would be implemented and as matter of configurability interleaving mode is also incorporated.

Moreover, 6x6 interleaving switch-based crossbar architecture along the proposed mechanism is implemented and verified on Xilinx FPGA Virtex2P XC2VP30. The overall architecture primitive modules are designed, simulated, synthesized and structured in Verilog and by using various FPGA based EDA Tools.

Keywords: Round Robin, NOC.

Introduction

Today field programmable gate-arrays (FPGAs) are used for a wide sector of applications. The usage in former times was focused on rapid-prototyping system for integrating test systems. After the test-phase, often an ASIC approach substituted these systems for mass-production. Due to dramatic growth in circuit-design complexity regarding Moor's Law, the ability of implementing complex architecture in a single chip always presents new challenges. One of the issues found by designers while implementing large SoCs is the communication among their components. Buses are an increasingly inefficient way to communicate, since only one source can drive the bus at a time, thus limiting bandwidth.

NoCs are increasing in popularity because of their advantages: larger bandwidth, and lower power dissipation through shorter wire segments. Communications in large SoCs are so important that many designers have adopted the NoC approach. The challenges consist in offering the best connectivity and throughput with the simplest and cheapest architecture of methodology; whereas, many topologies and architectures have been investigated. This is well illustrated in [4], where researchers propose a two-level FIFO approach in order to simplify the design of the arbitration algorithm and improve the bandwidth. However, this method tends to be expensive in term of hardware.

Although the completely embedded tools of FPGA manufactures such as Xilinx and Altera are offered to help their customers to design the complex Multi Processor System on- Chip (MPSoCss), their environments only offer the bus based paradigm or point-to-point connection. More complex MPSoCs may require higher bandwidths than a bus-based system can offer, or may need to be more efficient than point to- point connections.

The idea of designing the Reconfigurable Crossbar Switch for NoCs to gain a high data throughput and to be capable of adapting topologies on demand was presented in [5]. Their evaluation results showed that output latency, resource usage, and power consumption were better than a traditional crossbar switch. Nevertheless, they did not focus the problem while operating many-to-one and one-to-many data communication.

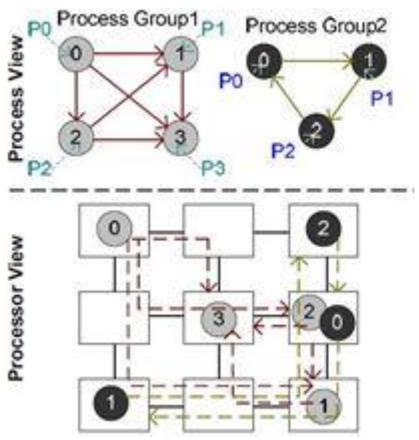
Our proposed crossbar architecture is designed with two contributions: First, our crossbar has to be light weight, requiring few FPGA resources, and thus suitable for both small and large resources with high bandwidth efficiency on FPGA based systems. Second, the proposed crossbar architecture hast o solve the output delay (output latency) problem while multicasting on all source situations. Moreover, in order to obtain the work efficiency, 6x6 interleaving switch-based crossbar is realized by Verilog, and verified on the Xilinx FPGA Virtex 2Pro XC2VP30. In respect of the verification, the test environment has introduced in this paper.

The rest of this paper is organized as follows: section 2 overviews communication problem. Section 3 presents switch based crossbar structure. In section 4, we implement the primitive component and 6x6 interleaving switch-based crossbar, and compare the proposed switch-based crossbar with general propose switch-based system and bus-based system.

Test environment and experimental result are presented in section 5. Finally, conclusion is summarized in section 6.

Communication Problem

In Fig. 1, all communication in a process group becomes many communications involving an arbitrary subset of processor from the system's point of view. In order to efficiently support data communication, a system has to support one-to-one (uni-cast), one-to-many (multicast), many to-one(gathering) and many-to-many communication primitives in hardware. Actually, most interconnection can support uni-cast, but not gathering and multicasting.



In Fig. 2(a) shows a multicast communication with three destinations where process P0 has to send the same data to three processors: P1, P2 and P3. Without the multicast functionality on interconnection, the system can use unicast functionality to send data to all destinations sequentially. However, blocking will occur, if P0 is executing sending state to P1, and P1 has not yet execute receiving state, P0 is blocked; meanwhile, P2 is executing receiving state, and is blocked because P0 has not yet executed sending state. Obviously, system resources are wasted due to unnecessary blocking. Fig. 2(b) shows gathering data communication from three sources to one destination. Because of sending data from all sources, congestion is found at destination where it can be reduced by applying source priority. Supposing P1, P2 and P3 are the first, second and third priorities. While their data arrive at P0, the data from P1 will be the first forwarding; meanwhile, the rest will be buffered. Thus, the output latency of system will increase, and buffer will also require.

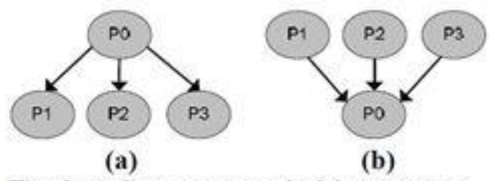


Fig. 2. a) One-to-many b) Many-to-one

In order to solve these problems, we apply interleaving technique to switch-based crossbar architecture on FPGA, and compare resource usage as well as estimated frequency with [1,2].

Switch-Based Crossbar Structure

The major components that make up the switch-based crossbar consist of Input Port module, Switch module and Output Port module. Depending on the kind of channels and switches, there are two alternatives for designing switch-based crossbar: unidirectional and bidirectional [6]. In this paper, unidirectional switch-based crossbar is selected and simplified with 6x6 switch-based crossbar structure shown in Fig. 1 because of resource constraint.

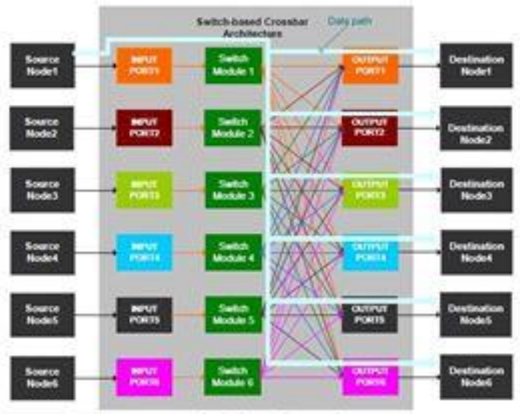


Fig. 3. 6x6 switch-based crossbar structure

Obviously, transmitting data from a Source Node to a Destination Node require crossing the link between the Source Node and the Input Port module, and the link between the

Output Port module and the Destination Node where the Switch module in data path will dynamically establish the link for the Output Port

ISSN: 2277-9655
Impact Factor: 1.852

module according to switching protocol. According to the 6x6 switch-based crossbar architecture, its switching protocol shows in Fig. 2.

Because of resource constraint on the target FPGA [2], data width, the N.of word register, the destination port register are 16,10 an d 6 bits respectively. Moreover, synchronous protocol is applied to synchronize all modules in the switch-based crossbar-Clock signal, Ready signal and Acknowledge signal.

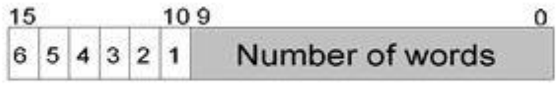


Fig. 4. 16-bit switching protocol

Fig. 2 shows the 10-bit Last Significant Bit (LSB) of switching protocol, which defines the number of packets required transferring from a Source Node to a Destination Node, where the maximum is 1,024 packets per time, and the rest define the Destination Node. For example, when the Source Node1 wants to transfer 100 packets to the Destination Node number 3 and 4, the 16-bit switching protocol has to be 0011_0000_0110_0100 (0x3064H).

A. Input Port module

In this section, the architecture and behavior of the Input port module are detailed. As shown in Fig. 5(a), there are three components in this module comprised of a 16-bit tri-state buffer, a 1-bit tri-state buffer and a finite state machine (FSM) component where the N.of word register and the destination port register are resided. Its behavior shows in Fig. 5(b) based on switching conceptual [1].

At the beginning, the state is in an idle state, and then the header ready signal enables HIGH to inform a Source Node that ready to read the switching protocol. As soon as the 16-bit switching protocol is asserted at the

Data in signal and the Data in valid signal, the state goes to the next check state. In this state, the 16-bit switching protocol is separated and written on the N.of word register and the destination port register resided in FSM module, where 10-bit low and 6-bit high are written on the N.of word register and the destination port register; meanwhile, the register port signal is read to check, whether or not the required destination ports are free. If it is free, the state will go to the req state.

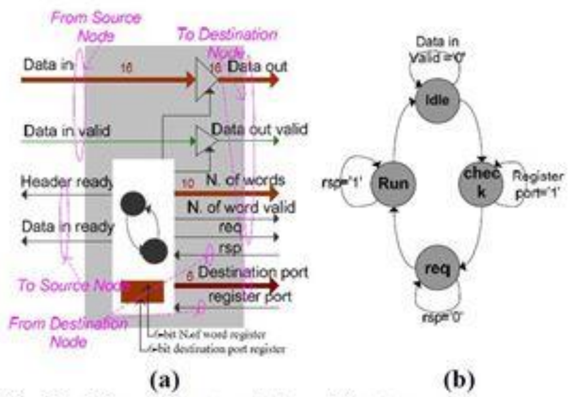


Fig. 5. a) *Input Port module* architecture
b) State-machine

In the req state, the N.of words register, the destination port register and the req signal are read and presented on the N.of words signal and the N.of word valid signal. When the rsp signal enables HIGH, the channel for transferring data is guaranteed, and the state goes to the Run state. In the Run state, the data in ready signal enables HIGH. The 16-bit data in signal and the data in valid signal are sequentially asserted into the channel. When the lasts data is completely forwarded, the rsp signal from the Output port module will enable LOW, and the state will start to the next cycle.

B. Output Port module

In this section, the Output Port module architecture and the flow diagram are explained in detail. As shown in Fig. 6(a), there are three components of this module, comprised of a 16- bit six-port multiplexer component which used to multiplex the Data in signals coming from the Switch Module, the Round- Robin component assigned to guarantee the fairness of all input requirements and the FSM component where the 10-bit counter register and the 6-bit Port Status register used to count down a number of through packets and to identify that can occupy the path of this Output port module are resided. Since interleaving mechanism bases on Time-Division-Multiplexer (TDM)[7], the 16-bit six-port multiplexer component can multiplex all Data in signals depended on Time Slot. For instance, Fig. 7(a) and 7(b) show the timing diagram, and the 6-bit Port Status register where all Source node and the Source Node number 1,2,5 and 6 require transferring data to one Destination Node, the 6-bit Port Status register in the Destination Node will be 111111 and 110011.

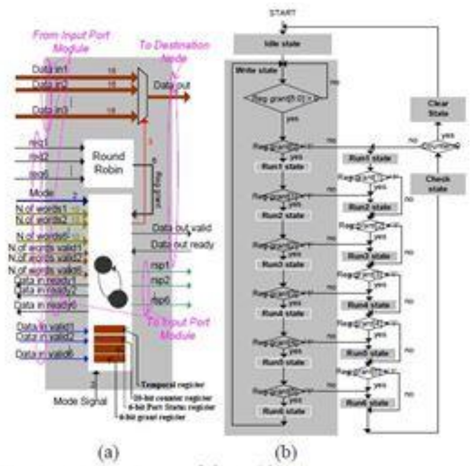

(a) (b)
Fig. 6. a) *Output Port module* Architecture
b) Flow control diagram

Fig. 6(b) explains its behavior. At first, the state is in an Idle state, and the mode signals are read where there are two kind of possible modes, normal mode and interleaving mode. In the normal mode, the Round Robin component will be enabled but will be disabled in the interleaving mode. Whenever a Source Node requires transferring its data to a Destination Node, the req signal of the Input Port module connected with the Source Node will enable HIGH. Then, at the Output-Port module connected with theDestination Node, each bit of the 6-bit grant register related with each req signal of each Input Port module will set "1". The N.of words signal and the N.of words valid signal will be read and stored into the temporal register. Then, the state goes to the Write state. In the Write state, each bit of the grant register is read to check the Source Node's requirement and to identify the location of the next state. In order to simplify, supposing the grant register contains 101100.

It implies that the input port modules number 3,4 and 6 will forward their data to the output port module. Therefore, the state will start at the Run3 state, and the Data in ready signals number 3,4 and 6 will be enabled HIGH. Until the Run6 state is executed, the state goes to the Check state. The counter register counting the forwarded packets is checked, whether or not the forwarded packets has been completely sent. Whenever, they have been sent properly this counter register will be "0000000000" and the Clear state will be done. At the Clear state, the rep signals at 3,4 and 6 are enabled LOW and the grant register will be"000000", as well as the state will begin the next cycle

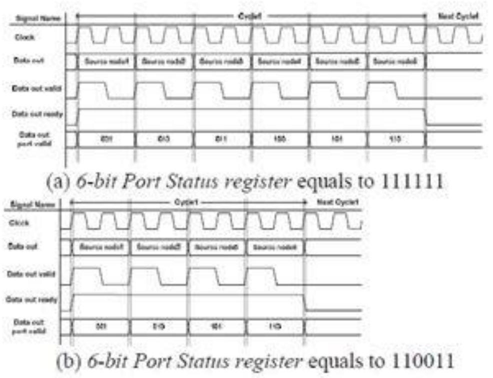

(a) *6-bit Port Status register* equals to 111111
(b) *6-bit Port Status register* equals to 110011

Fig. 7. Timing diagram a) 6-port to one-port b) 4-port to one-port

C. Switch Module

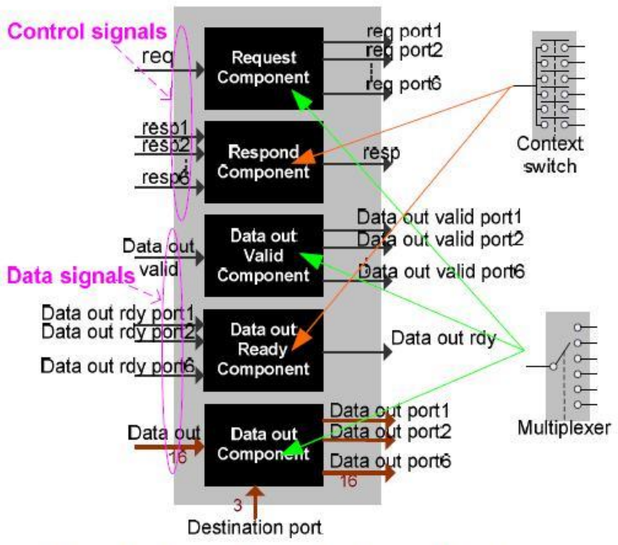
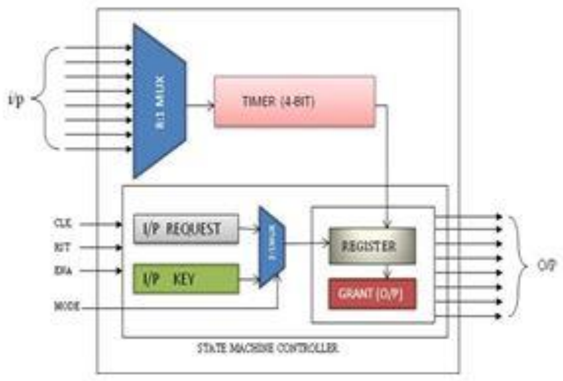


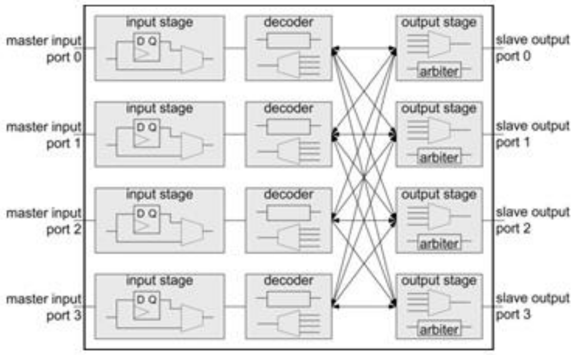
Fig. 8. *Switch module* architecture

Switch module crosses the link between the Input Port module and the Output Port module. Underneath the Switch module, there are five components connecting with two sets of signal. First, the control signal consists of the req signals and the resp signals. Second, the data signal composes of the data out valid signal, the data out rdy port signals and the data out signal.

Two types of switch component are context switch and multiplexer. All components ofeach set can be mapped with two types of switch component as shown in Fig. 8, where they are controlled by the 3-bit Destination port signals of Input Port module



The main design modules consist of interleaving timer module, Request register, data-mux, Mask register and state Machine Controller and Grant



The designed arbiter project is proposed in AMBA-AHB Matrix.

Implementation

The problems in section 2 and the primitive modules of the switch-based crossbar architecture in section 3 are implemented and realized on the target FPGA in this section. All primitive modules are structured by Verilog, verified their behaviors with Model Sim 6.2c, synthesized their resource usages and estimated frequency on the Xilinx FPGA Virtex 2P XC2VP30 by ISE tool as shown in Table1. To realize the target FPGA, 6x6 interleaving switch-based crossbar is introduced.

Test Environment And Experimental Result

The 6x6 interleaving switch-based crossbar designed and implemented in section 4 is evaluated on test environment as shown in Fig. 9. The Source Nodes and the Destination Nodes connecting with our crossbar can reconfigurable where several IP cores can take place. Before testing, 16-bit counter modules are placed on all Source Nodes, and the available FPGA I/O pins are mapped to all Destination Nodes.

The environment is set as following:

1. The system operates at 100 MHz,
2. The interleaving mode is set to default,
3. All 16-bit counters are ready to generate the switching protocol and the 256 packets,
4. TLA5204 logic connects to this environment to measure and capture the 16-bit data out signal and data out valid signal at the Destination node.AHB Matrix. module.

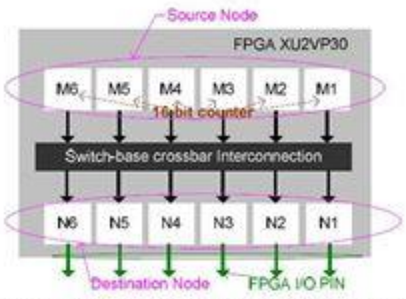


Fig. 9. Test environment on Xilinx FPGA

Since the bandwidth and time period are considered, when more than one *Source Nodes* require transferring a large number of packets to the same *Destination Node* simultaneously, the number of the *Source*
Node is increased one at a time in test cases. Fig. 10 and 11 show the captured data while one *Source Node* and six *Source Nodes* are transferring the 256 packets to the same *Destination Nodes* with interleaving functionality on our crossbar

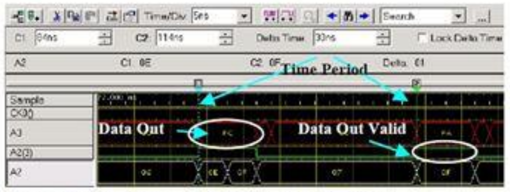


Fig. 10. One-to-one data communication

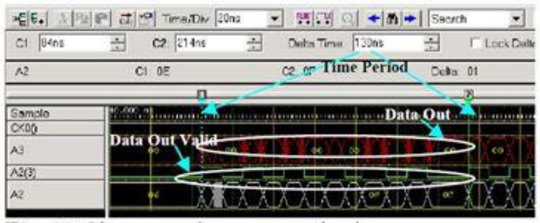


Fig. 11. Six-to-one data communication

The bandwidth results come out from this formula:

*BW = Nof Packet *bit * Nof Inputdata /TimePeriod*

where *BW* is bandwidth, *NofPacket* is the number of packets, *bit* is data width, *N.ofInputdata* is the number of input data, and *Time period* is the time period of *data out signal* measured at the *Destination Node*.

ISSN: 2277-9655
Impact Factor: 1.852

Conclusion

This paper proposes and implements the interleaving mechanism to overcome the output delay (latency) while operating many-to-one and one-to-many data communication. Moreover, resource usage and bandwidth are acceptable when it is compared with the general propose switch-based system and bus-based system.

References

[1] D. Bafumba-Lokilo, Z. Savaria, J. David ,"Generic Crossbar Network on Chip for FPGA MPSoCs", Circuits and Systems and TAISA Conference, 2008, pp 269-272.

[2] Xilinx , "On-chip Peripheral Bus V2.0 with OPB arbiter", "Processor Local Bus(PLB)v4.6(v1.03a)", http://www.xilinx.com.

[3] H.Po-Tsang, H. Wei , "2-Level FIFO Architecture Design for Switch Fabrics in Network-on-Chip", Circuits and Systems, ISCAS 2006, Proceedings 2006, IEEE International Symposium on, pp.4863-4866.

[4] M. Hübner, L. Braum,D. Göhringer, J. Becker, "Run-Time Reconfigurable Adaptive Multilayer Network-On-Chip for FPGA-Based systems", IEEE International Symposium on In Parallel and Distributed Processing, 2008, pp. 1-6.

[5] C. Hilton, B. Nelson, "PNoC: a flexible circuit-switched NoC for FPGA based systems", IEE Proceeding Computing Vol. 153, 2006, pp. 181-188.

[6] C. Qiao, R.Melhem, "Reconfiguration with Time Division Multiplexed MIN's for Multiprocessor Communication", IEEE Transactions Parallel and Distributed System, Vol. 5, 1994, pp. 337-352.