Synchronous FIFO Buffer Model Design for Network on Chip

Shilpa Kodgire
Department of Electronics Engineering
NIELIT, Aurangabad, Maharashtra, India

Dr. U.D. Shiurkar
Department of Electronics and Telecommunication Engineering
DIEMS, Aurangabad, Maharashtra, India

Abstract- As per moors law, in every second year "The scaling of chip technologies has led to a doubling of available processing resources on a single chip. Because of the continuous efforts of scientists the continuous innovation in semiconductor technology is taking place which enables more complex System on-Chip (SoC) designs on the same constraints. Many more intellectual properties (IPs) are integrated into a SoC to provide various functions, including communications, networking, multimedia, storage, etc. A chip constitutes an entire system, opens up the feasibility of a wide range of applications making use of massive parallel processing and tightly interdependent processes. The Network-on-Chip (NoC) approach was introduced to handle communication aspects of chip design, due to its better scalability, flexibility, energy efficiency, testability and differentiated services while hierarchical system integration using multiple smaller buses connected through repeaters or bridges offer possible solutions, such approaches tend to be ad-hoc in nature, and therefore, lack generality and scalability. The bus-based scheme still fails to satisfy the requirements of future SoC mainly due to two major drawbacks. Non-scalable and the bandwidth are shared by all IPs and thus the bus becomes the performance bottleneck when more and more IPs is connected. In order to interconnect such a high number of elements on a die, researchers have turned to Network on Chip as a replacement to conventional shared buses and ad-hoc wiring solutions. Performance evaluation of the routing node in terms of latency is the characteristics of an efficient design of Buffer in input module. The utilization efficiency of the packet buffer array improves when a common buffer is used instead of individual buffers in each input port which propose the use of multi-port multi clock buffers such as Dual clock Dual Port Buffers on First in First out (FIFO). It has been observed that latency in Common Buffer improved by 46% over its distributed buffer.

Keywords –Processing Elements (PE), service rate, FIFO, latency, packet array, IP mapping.

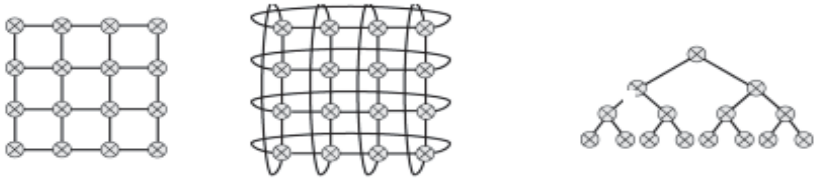
I. INTRODUCTION

The scaling of microchip technologies has led to a doubling of available processing resources on a single chip every second year. Even though this is projected to slow down to a doubling every three years in the next few years for fixed chip sizes. Though the evolution is continuous, the system level focus, or system scope, moves in steps. When a technology matures for a given implementation style, it leads to a paradigm shift. SoC opens up the feasibility of a wide range of applications making use of massive parallel processing and tightly interdependent processes, some adhering to real-time requirements, bringing into focus new complex aspects of the underlying communication structure. Many of these aspects are addressed by NoC. Modules on a chip, such as processors and memories, are traditionally connected via a single shared link (a bus). As chips become more and more complex, and the number of modules on a chip increases, this bus architecture becomes less efficient because a bus cannot be used by multiple modules simultaneously. Networks on chips are an emerging paradigm for the connection of on-chip modules. In networks on chips, data is transmitted by packet switches, so that multiple links can be used at the same time and communication becomes more efficient. These switches have buffers, which leads to many performance-analytic questions that are interesting from both a theoretical and practical point of view. Networks on chips use switchesto transmit data across the network. A switch consists of input and output ports. Data packets arrive to the input ports of the switch, and leave from the output ports. If multiple input ports have data for the same output port, only one input port can transmit its data, and the switch selects only one. Data that is not transmitted immediately is stored in buffers and will be transmitted later.

NoCs must offer quality-of-service (QoS) guarantees on the latency, jitter, and throughput of certain classes of network traffic. Meeting these constraints depends on spatial and temporal properties of traffic patterns. Inter-node communication in a NoC is performed by the transmission of data packets through routers. Each packet is communicated as a number of smaller units called flits. Store and forward networks wait until all flits of a packet are received before forwarding any to the next node, while wormhole networks forward the individual flits of a packet as they arrive. Wormhole networks can have lower latency and use smaller buffers than store and forward [15], but a single packet can be strung out across the network blocking the forward progress of many other flows and complicating analysis [1]. Adding virtual channels allows flows to be routed around any backed-up flows.

The network is dedicated to delivering messages from source node to destination node (BJE, 2006). The most important factors that have to be defined in network layer are the topology of the network and the transmission protocol that it uses. Topology determines the connections between nodes on chip. The most common regular topology is the mesh, in which every node connects to its closest neighbors. Other regular topologies the next thing needed for the right operation the network layer is the protocol. The protocol determines the path that data should follow, from source to destination node.

Designer of the network protocol can be chosen between: (a) dedicated paths for each route (connection oriented) or dynamic forwarding from source to destination (connectionless) (b) routing of packets according to predefined routing strategies (deterministic routing) or dynamic routing by checking the availability of links in time of transmission (adaptive routing) (c) control of data flow, that can be done locally on each node or globally by one or more dedicated nodes (d) shortest path routing or not (e) delayed arrival or rejection of a packet after a number of hops and others. In this work, the topology chosen is irregular and minimal routing with connectionless links and locally control, with local routing tables in every node, is implemented.



a) Mesh b) Torus c) Binary Tree

Figure 1.Different Topologies used in NOC

A packet-switched 2-D mesh is the most used and studied topology so far. It is also a sort of an average NoC currently. Good results and interesting proposals provoke design engineers to use this topology as the base [1].The key research problems in the design of NOCs include but are not limited to topology, channel width, buffer size, floor plan, routing, switching, scheduling, and IP mapping [2].

Additionally, [3] lists research issues to be application modeling and optimization, NoC communication architecture analysis and optimization, NoC communication architecture evaluation, and NoC design validation and synthesis. The most important metrics for NoCs are application runtime, silicon area, power consumption, latency and throughput. All these are to be minimized and usually appropriate trade-off is sought[4]**.** The required silicon area is the most commonly reported value (77%) followed by latency (55%) and maximum operating frequency (50%). The other metrics have lower occurrence[1].

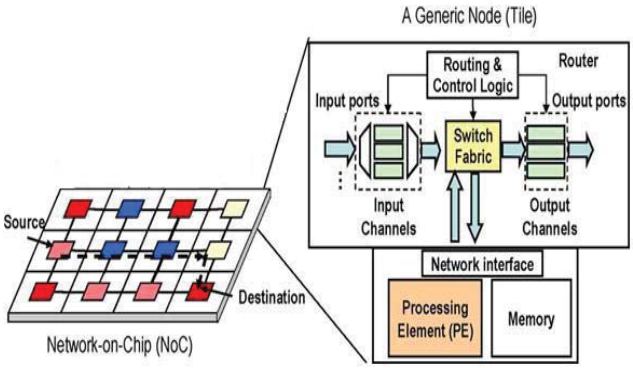


Figure 2. Network on chip Architecture

II.PROCESSING ELEMENTS AND ROUTERS

Processing elements are the generators of traffic inside the NoC. Every processing element transmits packets according to its packet injection rate, which can be defined by the user as a command line option. The traffic is generally random, if not defined otherwise, produced by a pseudo-random generator. However, user is free to define exactly the traffic produced by PEs. This can be done by writing a traffic table, in which every line gives all the necessary information for a single transmission.

Source	Destination	Packet injection rate	Probability of retransmission	Time in cycles transmission begins	Time transmission ends	Period between two successive transmissions

Figure 3.Traffic Table

Every PE has a port for clock signal and one for reset signal and connects to one router by using a group of ports, Routers are the most important components, referring to communication of NoC. Their objective is to receive packets, read the appropriate fields on head flit of every packet, decide the next hop, check availability of links, reserve links and forward packets. Routers also have reset and clock ports. In the default version of tool every router was connected to one local PE and four neighbors in directions north, south, east and west with an input buffer for every connection.

NoC architectures are based on Packet switch network, which is the new and efficient principle for router design. In packet switched network some amount of buffering is needed. Buffers can be provided both at input and output. In this paper the routing algorithm used is X-Y routing. This algorithm is more area and power efficient.

III. DESIGN METHODS IN FIFO

FIFO can be designed in two domains

- Single clock domain
- Multiple clock domain

Depending on the clocking frequency which is getting used in the systems FIFO is categorized as

- Synchronous FIFO
- Asynchronous FIFO

A. Synchronous FIFO

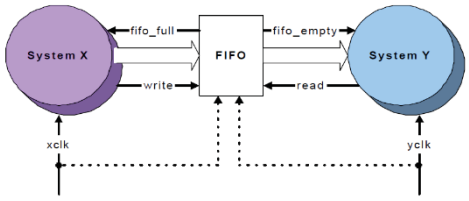


Figure.4. Block Diagram ASDCFIFO

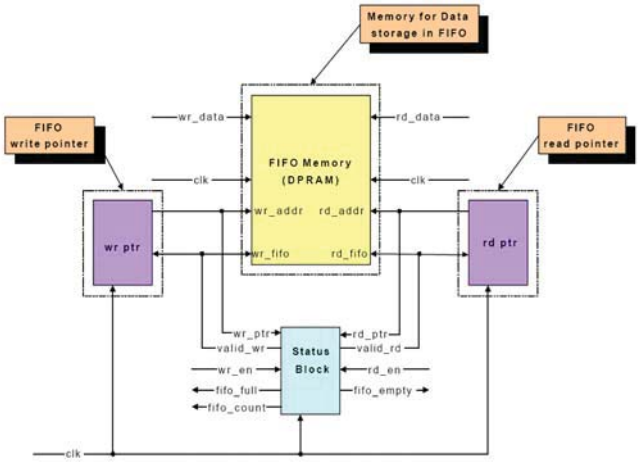


Figure 5. Block Diagram of Synchronous FIFO Architecture

B.Working of synchronous FIFO

Reading and writing operation in synchronous FIFO is done on the same clock. Figure 5 shows a general architecture of a Synchronized FIFO. DPRAM (Dual port RAM) is used as a FIFO memory since we require reading and writing to be independent. The read and write ports have separate read and write addresses generated by two read and write pointers. The write pointer points to the location that will be written next and the read pointer to the location that will be read next.

A valid write enable increments the write pointer and a valid read enable increments the read pointer. Shown above in the figure, is a status block that generates the "fifo_empty" and "fifo_full" signal. If "fifo_full" is asserted it means that there is no room for more data to be written into the FIFO. Similarly "fifo_empty" indicates that there is no data available in the FIFO to be read by the external world. This block also indicates the number of empty or full locations in the FIFO at any point of time by performing some logic on the two pointers.

The DPRAMabove can have either synchronous reads or asynchronous reads. For synchronous read, an explicit read signal is supposed to be provided before the data at the output of the FIFO is valid. For asynchronous reads, DPRAM does not have registered outputs; valid data is available as soon as It is written (data is read first and then the pointer is incremented).

On reset both the read and write pointers are '0'. "fifo_empty" signal is asserted and "fifo_full" remains "low" during this time. Further reads from FIFO are blocked when FIFO is empty so only write operation is possible. Now subsequent writes on the FIFO increments the write pointer and desserts the FIFO empty signal. A point is reached where there is no room for more data when the write pointer equal to RAM_SIZE –1 .At this time a write causes the

write pointer to again roll back to '0' and "fifo_full" signal is asserted. Looking above we have the same condition when the FIFO is empty or full that is when read pointer equals to the write pointer. It is necessary to distinguish between these two conditions.

C.FIFO full and FIFO empty generation:

All the transitions shown in above figure are in subsequent clocks. As shown above in the figure if a write causes both the pointers to become equal in the next clock, then FIFO becomes full. So we have the following condition for assertion of fifo_full signal. When (read_pointer = write pointer + 1) and "write" à asserts the fifo_full signal When Write_ptr=Read_ptr+1, EMPTYnf=0 (i.e. FIFO is EMPTY)

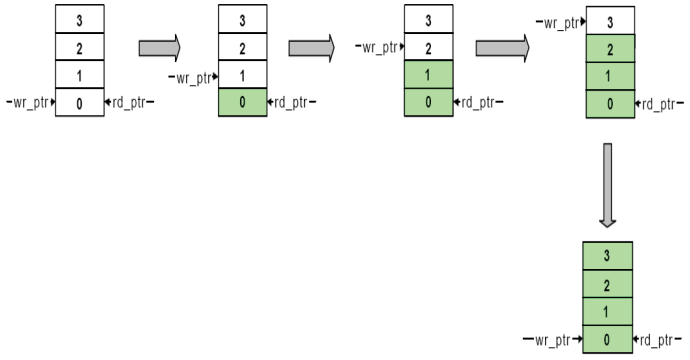


Figure6. FIFO full condition

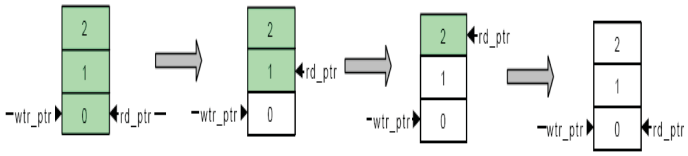


Figure7.FIFO empty conditions

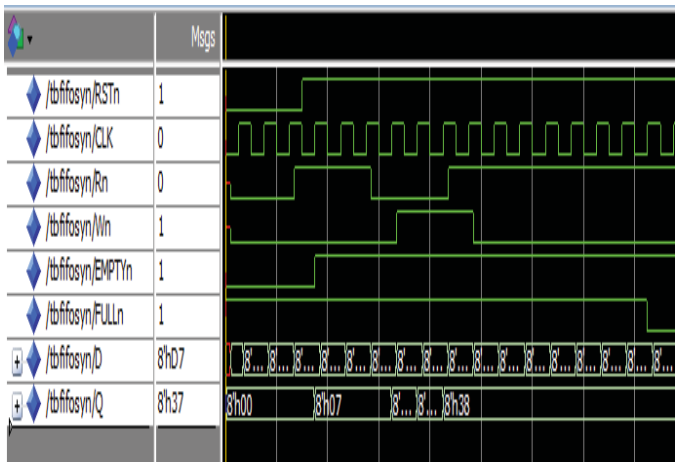


Figure 8.Waveforms of Synchronous FIFO

After RSTn is deasserted the write pulse Wn is asserted. Four data values 07, 0E, 1C and 38 are written into the data registers with the write pointers incrementing from 0 to 4. Note that his first value 07 has already shown up on the Q output. When read pulse is asserted, the read pointer is incremented by one and the next data is shown in the output Q. This show that data on the output bus is available to read. The read control circuit should get the data and assert the read pulse in the same clock cycle. Three data are read, the read pointer is pointing to the unread data register indexed three. Write pulse Wn is again asserted with seven data values 8D,1B,37,6E,DC,B9 and 72 written into data registers .we have written four values , read three values , and written seven values. All the data in the data registers remains unread. The FIFO is FULL and Full FLAG FULL_n is therefore asserted.

IV.CONCLUSION

We have designed Synchronous FIFO, and come to a conclusion that the time required to complete the operation of system in Synchronous FIFO's is very less as compared to other forms of FIFO.

REFERENCES

[1] "VLSI Architecture and FPGA Implementation of ICE Encryption Algorithm" a. p. fournaris, n. sklavos and 0. Koufopavlou electrical and computer engineering department University of Patras, Patras, GREECE ICECS-2003. 0-7803-8163-7/03/$17.00 0 2003 IEEE

[2] "First Step Toward Internet Based Embedded Control System". EkaSuwartadi, CandraGunawan, ArySetijadi P, CarmadiMachbub I Laboratory for Control and Computer Systems Department Of Electrical Engineering Bandung Institute Of Technology, Indonesia, 2004 5th Asian Control Conference.

[3] " Implementation of New Symmetric Ciphering on ATmega 32," Hussam M. Elbehiery and GhadaAbdelmouez M. 978-1-61284-185-4/111$26.00 ©2011 IEEE.

[4] "Custom Hardware Interface using NIOS II Processor through GPIO" Meghana A. Hasamnis and S. S. Limaye 978-1-4577-2119-9/12/$26.00_c 2011 IEEE.

[5] 'Low-Cost Advanced Encryption Standard (AES) VLSI Architecture: A Minimalist Bit-Serial Approach "Orlando J. Hernandez, Thomas Sodon, and Michael Adel 0-7803-8865-8/05/$20.00 ©2005 IEEE.

[6] "IMPLEMENTATION OF AES AS A CMOS CORE" Lan Liu and David Luke Department of Electrical and Computer Engineering University of New Brunswick Fredericton, NB E3B. CCECE 2003 - CCGEI 2003, Montreal, Mayimai 2003 0-7803-7781-8/03/$17.00 0 2003 IEEE.

[7] "IMPLEMENTATION OF AES AS A CMOS CORE Lan Liu and David Luke Department of Electrical and Computer Engineering University of New Brunswick Fredericton, NB E3B 5A3Canada CCECE 2003 - CCGEI 2003, Montreal, Mayimai 2003 0-7803-7781-8/03/$17.00 0 2003 IEEE.

[8] "Fast Implementation Of the Advanced Encryption Standard Using Atmega1281 for Sensor Nodes" Kasumi Toriumi, Yoshio Kakizaki, Keiichi Iwamura Department of Electrical Engineering Tokyo University of Science 1-14-6 Kudankita, Chiyoda, Tokyo 102-0073, JAPAN toriumi@sec.ee.kagu.tus.ac.jp.