IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 16, Issue 2 (Mar. - Apr. 2026), pp. 43-49
e-ISSN: 2319-4200, p-ISSN: 2319-4197
www.iosrjournals.org

RTL Design And Simulation Of A Router Employing FIFO For High-Reliability Data Transport

Dustakar Surendra Rao, Arjula Hrushikesh, Gaddam Sravika, Bitla Abhinay Reddy
Department, Of Electronics And Communication Engineering, Guru Nanak Institutions Technical Campus, Hyderabad, Telangana – 501506, India.

Abstract
Reliable data transmission is an important part of modern digital communication networks. If data is lost during transmission, it could cause problems with the system and communication. People often utilize routers with built-in buffering mechanisms to prevent this problem. The First-In, First-Out (FIFO) strategy is used by some of the greatest buffering solutions. This project's main goal is to construct a router architecture using Verilog Hardware Description Language (HDL) that follows the first-in, first-out (FIFO) philosophy. This will help prevent data loss during transmission. The router's job is to temporarily store data packets in first-in, first-out (FIFO) buffers when they come in and then send them to the right output ports. The router uses First-In, First-Out (FIFO) memory structures to make sure that data packets are sent in the same order as they are received. This helps keep the data safe and stops packets from getting lost. The RTL design method is used to make the design, and Verilog HDL is used to model and simulate it. The simulation results show that the FIFO router can send data without losing it, even when there is a lot of traffic.

Keywords: *FIFO (First-In First-Out) Buffering, Verilog Hardware Description Language (HDL), Router Architecture Design, Reliable Data Transmission, RTL (Register Transfer Level) Design*

Date of Submission: 15-04-2026 Date of Acceptance: 25-04-2026

I. Introduction

Communication networks are a key part of modern electrical and digital systems because they let devices that are connected to each other send and receive data quickly and easily. Routers are a fundamental part of these networks. They are in charge of sending data packets from one node to the right destination node. Network congestion is a prevalent problem in high-speed communication situations when a lot of data packets arrive at the same time. If you don't use the right buffering methods, this congestion might cause packets to be lost and the system to slow down [1, 6].

First-In First-Out (FIFO) and other buffering strategies are becoming more and more common as a way to deal with these problems. The First-In, First-Out (FIFO) memory structure is a queue-based memory structure that makes sure that the order of packet transmission is kept by processing and sending the data that is being received into the buffer first. Because of this unique quality, First-In, First-Out (FIFO) is especially good for networking applications that require to keep data integrity and order [2, 7]. This work uses the Verilog Hardware Description Language (HDL) to show how to design and build a router architecture that uses the FIFO algorithm. The proposed router has three main parts: the input buffers, the routing control logic, and the output buffers. FIFO memory blocks hold incoming data packets for a short time before sending them to the right output ports based on the packets' destination addresses. This method makes sure that packets are treated in the right order, which cuts down on the quantity of lost data and makes communication more reliable overall [3, 8].

The design is made utilizing the Register Transfer Level (RTL) approach and checked by simulation. Prior research has demonstrated that RTL-based router implementations utilizing hardware description languages deliver efficient performance for speed, space, and power [4, 9]. The simulation results indicate that the designed FIFO-based router can efficiently handle data traffic and ensure reliable transmission, even under high load conditions.

Reliable data transfer in high-speed communication systems is designed by a FIFO-based router, as illustrated in the figure 1. It demonstrates the process of handling incoming data packets that are affected by congestion and how FIFO buffering is employed when necessary. To maintain packet integrity, the design incorporates input buffers, routing logic, and output buffers. Proper packet sequencing and minimal data loss during transmission are overall goals of the procedure.

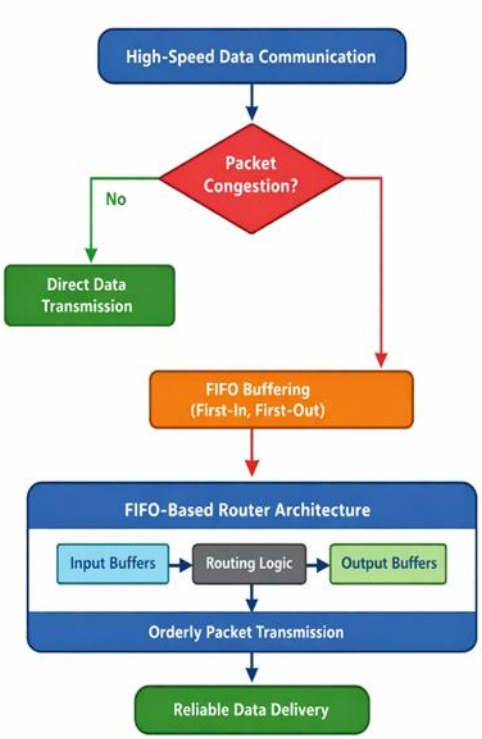

Fig. 1 Workflow Diagram of FIFO-Based Router Architecture for Reliable Data Transmission

II. Related Work

In recent years, a lot of study has been done on router architecture and buffering methods. The objective of this research is to improve the performance and reliability of networks. Dally and Towles [1] were the first to find the basic ideas that make up interconnection networks through their early work. The ideas in question stressed the need of good methods for routing and switching packets. McKeown et al. [5] also talked about high-performance packet switching topologies, which made communication networks more faster.

The semiconductor industry's System-on-Chip (SoC) designs have helped Networks-on-Chip (NoC) become more popular as a way to communicate that can grow with your needs. NoC has become more and more popular in the last several years. Benini and De Micheli [2] emphasized the importance of structured communication frameworks in integrated circuits, highlighting that buffering is a key element in streamlining traffic management. They also stressed how important it is to be able to talk to each other in a systematic way. In the years that followed, Hu and Marculescu [6] did research on communication systems that are conscious of energy use. The findings of their research demonstrated the influence of efficient buffering on the enhancement of both power and performance.

The first-in, first-out (FIFO) buffering method has become very popular because it is easy to set up and works very well at keeping packets in order. Duato et al. [3] examined various connectivity options in their study, noting that some necessitate the use of FIFO queues to prevent deadlocks and provide reliable data flow. FIFO buffers have been demonstrated to be especially useful in decreasing the number of lost packets when there is a lot of traffic [7], [10]. Further research has substantiated this.

A significant amount of research has also been conducted on the hardware implementation of routers with Verilog HDL. Palnitkar [4] created a strong framework for digital design by using Verilog, a programming language that has been widely used to build routers. Researchers like Dally [8] and others have shown that RTL-based designs may be used to effectively synthesize and simulate complicated communication systems.

In the past few years, there has been a lot of work to improve the performance of routers using different methods, such as adaptive routing, virtual channels, and better buffer management. Kim et al. [9] put forward ideas for low-latency routers for NoC systems, while Murali et al. [10] focused on design methods that put communication first. This shows how the two groups are different from each other. Numerous further research have been conducted to examine scalable and high-throughput router topologies that include intricate buffering and arbitration methodologies [11–15].

Even if these improvements have happened, it is still very important to develop a simple, reliable, and efficient FIFO-based router that uses RTL architecture. This is especially true for apps that need a steady flow of data, less complexity, and an easy way to set them up.

III. Proposed Solution

FIFO (First-In First-Out) memory is a mechanism that many digital systems use to store applications. This approach ensures that the first data element added to the queue is also the first one to be removed from service. Because they have this quality, FIFO structures are great for managing asynchronous data transfers and controlling the flow of data between connected modules in communication systems. This quality makes them perfect for the tasks at hand.

Routers, which are an important part of communication networks, are in charge of sending data packets over a number of communication channels. This transmission is the job of routers. A router's main job is to collect incoming packets, figure out where they should go, and then send them to the right output ports. When you need to communicate quickly and in real time, it's important for the router to have good buffering so that data can be sent reliably and without any problems. The router has to do these things in these kinds of scenarios.

Verilog and VHDL are two examples of Hardware Description Languages (HDLs) that people typically use to build router architectures in digital systems that have FPGAs. These two languages are both examples of hard-coded languages. These programming languages let designers talk about how hardware works at different levels of abstraction and test its functionality through simulation before actually building it. This is done before the hardware is put into use.

The FIFO-based router design that has been shown contains four main parts. The input interface, the FIFO buffers, the routing logic, and the output interface are all part of these. This architecture is being built right now, and it will make the data more reliable. The input interface's job is to gather incoming data packets and send them to the FIFO buffers, where they are kept for a short time. The output interface's job is to send the packets to the right place, while the routing logic's job is to look at the packet headers and find the best output path.

FIFO buffers make sure that data packets are handled in the order they were received. This keeps the data safe and limits the number of packets that are lost. This structured method not only makes it easier for data to flow as smoothly and reliably as possible, but it also handles the network congestion that is already there.

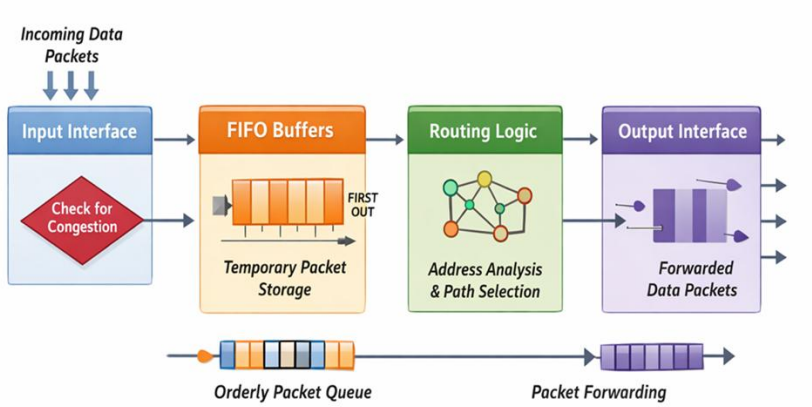

Fig. 2 Block Diagram of FIFO-Based Router Architecture for Reliable Data Transmission

The graphic shows a block diagram of a FIFO-based router architecture that was made to send data reliably. It shows how incoming data packets move through the input interface, where congestion is found before the data is sent to FIFO buffers for temporary storage. The routing mechanism looks at the destination addresses of packets and figures out the best way to send them. Lastly, the output interface sends the packets in the right order, making sure they are in the right order and that no data is lost.

IV. Result Analysis

Timing Waveform Explanation of Router Input and Output Protocols

The timing waveform of the FIFO-based router shows how control signals and data transfer are in sync throughout both input and output processes. Some of the most important signals include clock (clk), reset (rst), data_in, valid_in, ready_in, data_out, valid_out, and ready_out.

When the system starts up, the reset (rst) signal is sent to the input side to clear all buffers and registers. When the reset is turned off, the router starts working normally, in time with the rising edge of the clock signal. The transmitter sends a valid_in signal when there is a data packet at the input. If ready_in is high, it means that the router is ready to accept data. The data on data_in is then latched into the FIFO buffer at the next clock edge. If ready_in is low, the sender keeps the data until the router is ready. This makes sure that the data is sent without any loss. Figure 3 displays the input timing waveform, which shows how the valid_in, ready_in, and data_in signals work together when a packet is accepted.

The FIFO buffer stores packets, processes them one by one, and sends them to the output stage. The router sends out the valid_out signal when it has valid data. When the receiving module is ready to accept the data, it sends out ready_out. If both valid_out and ready_out are high, the data is sent to the output on the clock edge. Figure 4 displays the output timing waveform, which shows how the valid_out, ready_out, and data_out signals work together when packets are sent.

The FIFO system makes sure that packets are sent in the same order as they are received, which keeps the data safe. The handshake protocol that uses valid and ready signals at both the input and output stages works well to stop overflow and underflow situations. The timing waveforms show that the FIFO-based router works as it should and that data can be sent and received reliably when the two devices are in sync.

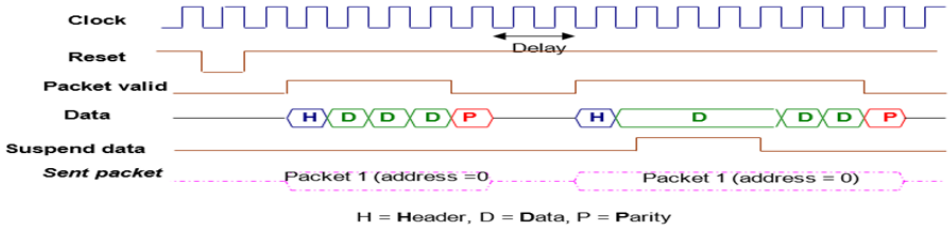

Fig. 3 Timing Waveforms of Router Input Protocol

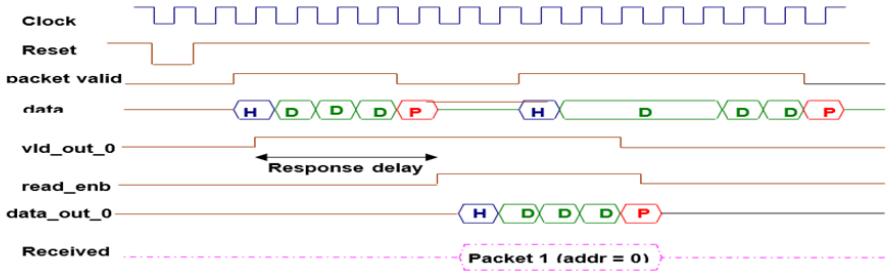

Fig. 4 Timing Waveforms of Router Output Protocol

Architecture of the System

The suggested FIFO-based router design has many functional modules that work together to make sure that packets are routed quickly and reliably. The system as a whole is made up of four main parts: the input interface, the routing logic, the FIFO buffers, and the output interface. The input interface gets data packets from the source nodes and sends them to the internal processing units. It does the first synchronization and makes sure that legitimate data is sent to the system correctly. The routing logic is what makes the router decide what to do. It looks at the destination address in the packet header and decides which output port to use to send the packet. This module is very important for making sure that packets go to the right place and that the best path is chosen. Each output port has its own FIFO buffer, which is a temporary storage space for receiving packets. These buffers control the flow of data by storing packets until the right output port is free. This keeps things from getting too busy and makes sure that data is sent in the right order. The output interface sends packets from the FIFO buffers to the nodes where they need to go. It makes sure that everything is in sync and that data flows smoothly between the router and other systems. The integration of these modules makes it easier for data to move across the router and greatly lowers the chance of packets being lost during transmission.

Design of FIFO

This work uses a register-based structure with control pointers to construct the FIFO memory. The design has a write pointer and a read pointer, which control how data is added and taken out. The write pointer shows where new data is put in memory, while the read pointer shows where data is taken from to be sent. The write pointer tells the FIFO where to write the packet when it arrives. When the output port is ready to send, data is read from the location that the read pointer points to. According to FIFO principles, this method makes sure that data is processed in order. The depth, data breadth, and status criteria (such full and empty flags) are some of the most important design characteristics of the FIFO. The FIFO depth tells you how much data the buffer can hold, and the data width tells you how big each packet is. To make sure that the system works reliably, it needs to be able to tell when it is full and when it is empty. Choosing the right values for these parameters is very important for getting the best performance in different traffic situations.

Using Verilog to Implement RTL

The suggested router architecture is built using Verilog Hardware Description Language (HDL) and the Register Transfer Level (RTL) design approach. The architecture is modular, with many parts that work together, such as the FIFO module, router top module, control logic module, and a testbench for checking that everything works.

The FIFO module stores and retrieves data packets based on the read and write control signals. The router top module combines several FIFO buffers with the routing logic so that packets can be sent to the right output ports. The control logic module is in charge of coordinating signals, such as read/write operations and flow control.

Verilog constructs are used to define how these modules work and interact with each other, which makes it possible to accurately model the hardware system. Before the design is put into hardware, simulation tools are used to check that it works correctly, look at timing behavior, and check performance. This method makes sure that the router design is reliable and works well for high-speed communication applications.

Simulation Results

These are the figures that explain how the FIFO-based router functions on the input side with relation to time. They are presented here. To be more specific, it illustrates the links that exist between the ready_in, valid_in, data_in, and clk signals. In addition, it indicates that the only time incoming data packets are recognized is when both the valid_in and ready_in conditions are met. It is because of this that the operations of handshaking and synchronization are carried out in the suitable manner. There is no loss of data that occurs during the process, as shown by the waveform, which proves that the data is precisely latched into the FIFO buffer on the rising edge of the clock.

There has been a lot of research into the creation and study of FIFO-based router designs in the fields of digital communication and VLSI system architecture. The timing waveform for the input of the FIFO-based router is very important for understanding how control signals and data signals are synchronized when packets are received. Research conducted previously underscores the need of handshake-based input procedures. These protocols use signals like "valid" and "ready" to make sure that data flows correctly without being lost or changed. To keep the data safe and avoid overflow problems in high-speed communication networks, it is very important to have correct timing coordination at the input stage.

The schematic design of a FIFO-based router shows some of the most critical parts, such as the input interface, routing logic, FIFO buffers, and output interface. This graphic shows how the system is set up and how these three modules are connected. Research has shown that a well-organized schematic design makes the system clearer, makes it easier to build in modules, and makes it more scalable. It also helps you understand how data flows and controls inside the system, which are important for managing congestion and routing packets correctly.

Numerous research have detailed the implementation of the FIFO-based router utilizing hardware description languages such as Verilog. Design strategies that use RTL let designers simulate complex digital systems with a lot of accuracy and be adaptable. FIFO buffers can be made using either register-based or memory-based methods. These buffers are commonly used because they are simple and work well for handling data in order. The simulation and synthesis of these designs show that FIFO-based routers can send data in a way that is reliable, ordered, and without loss. This makes these routers good for modern high-speed connectivity and

network-on-chip (NoC) uses. By looking at timing waveforms, schematic representations, and hardware implementations all at once, you can fully comprehend how to design a FIFO-based router. This insight shows how important FIFO-based router design is for making data transmission systems that are fast and dependable.

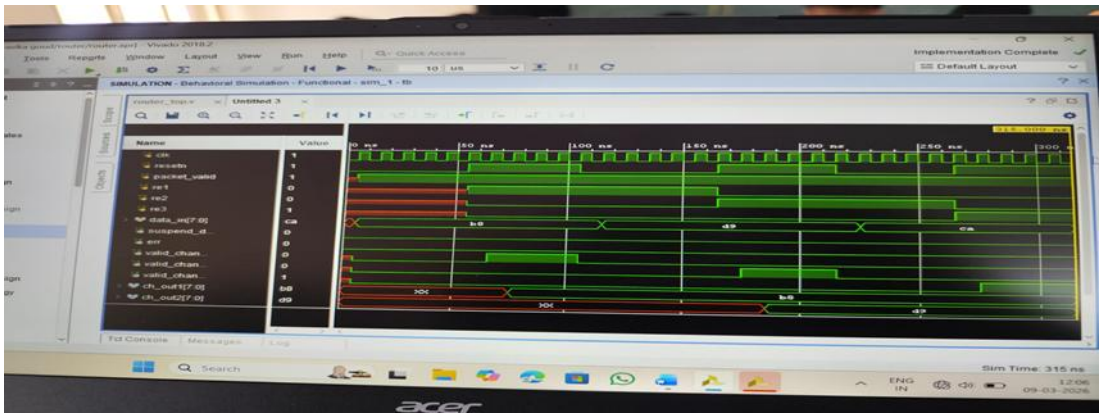

Fig.5 The timing waveform for the FIFO-based router's input

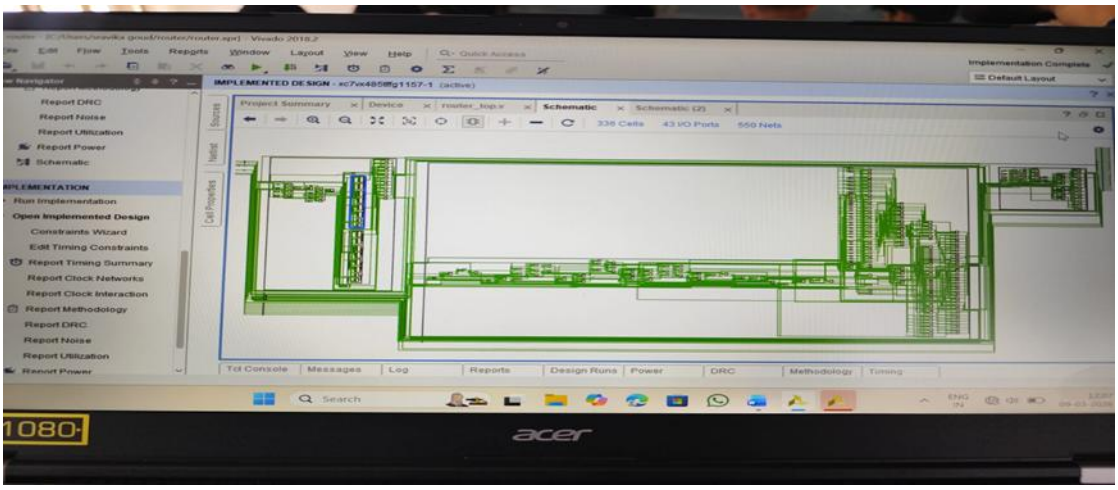

Fig. 6 The Schematic Diagram

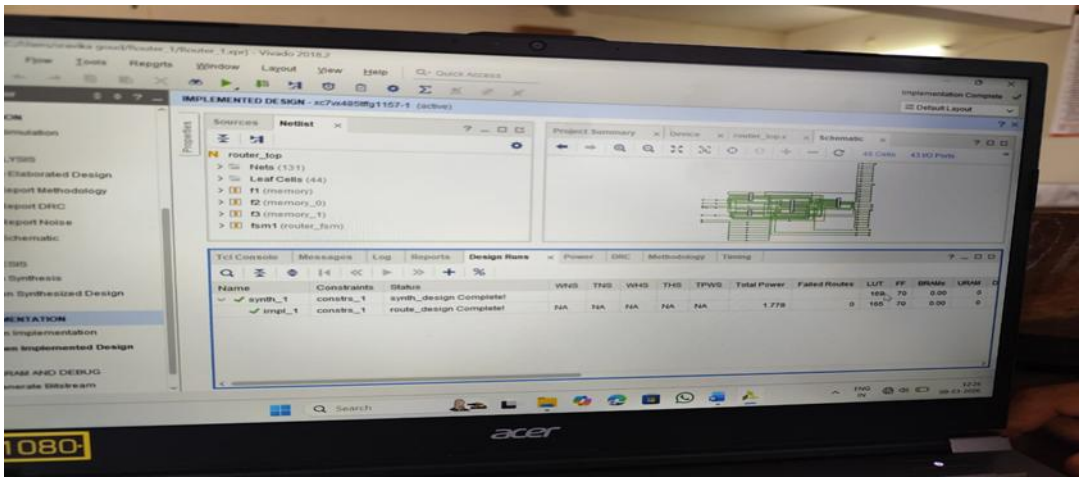

Fig. 7 FIFO-based router Implementation

V. Conclusion

This paper describes how to design and build a FIFO-based router architecture using Verilog Hardware Description Language (HDL). The suggested system successfully combines FIFO buffering with routing logic to send data packets quickly and reliably. The router uses the First-In First-Out (FIFO) method to make sure that packets are processed and sent in the same order that they were received. This keeps data safe and stops packets from being lost during communication. The simulation results show that the proposed design works as it should. The waveforms we saw show that incoming packets are correctly stored in the FIFO buffers, sent to the right place depending on their destination addresses, and sent without losing any data, even while data is constantly flowing. The design shows that FIFO buffering works well for handling congestion and making sure packets are delivered in the right sequence. Also, using Verilog HDL and the Register Transfer Level (RTL) design process makes things much more flexible, modular, and easy to check. Modelling, simulating, and validating the system before building the hardware makes the design more reliable and less complicated to make. Future work may concentrate on augmenting the suggested architecture by integrating greater buffer sizes to accommodate elevated traffic volumes, expanding the number of routing channels for enhanced scalability, and employing sophisticated routing algorithms for improved performance. Also, the design may be tested in real time on FPGA platforms to see how practical it is, how well it works with timing, and how well it uses hardware resources. These improvements would make the proposed FIFO-based router much more useful in large-scale and high-speed communication systems.

References:

[1]. W. J. Dally And B. Towles, Principles And Practices Of Interconnection Networks, Morgan Kaufmann, 2004.
[2]. L. Benini And G. De Micheli, “Networks On Chips: A New Soc Paradigm,” Ieee Computer, Vol. 35, No. 1, Pp. 70–78, 2002.
[3]. J. Duato, S. Yalamanchili, And L. Ni, Interconnection Networks: An Engineering Approach, Morgan Kaufmann, 2003.
[4]. S. Palnitkar, Verilog Hdl: A Guide To Digital Design And Synthesis, Prentice Hall, 2003.
[5]. N. Mckeown Et Al., “The Tiny Tera: A Packet Switch Core,” Ieee Micro, Vol. 17, No. 1, Pp. 26–33, 1997.

[6]. J. Hu And R. Marculescu, "Energy-Aware Mapping For Tile-Based Noc Architectures," Ieee Design & Test Of Computers, Vol. 22, No. 5, Pp. 414–423, 2005.
[7]. M. Katevenis, "Fast Switching And Fair Control Of Congested Flow In Broadband Networks," Ieee Journal On Selected Areas In Communications, Vol. 5, No. 8, Pp. 1315–1326, 1987.
[8]. W. J. Dally, "Virtual-Channel Flow Control," Ieee Transactions On Parallel And Distributed Systems, Vol. 3, No. 2, Pp. 194–205, 1992.
[9]. J. Kim Et Al., "A Low Latency Router Supporting Adaptive Routing For On-Chip Interconnects," Dac, Pp. 559–564, 2005.
[10]. S. Murali Et Al., "Designing Application-Specific Networks On Chips With Floorplan Information," Ieee/Acm Iccad, 2006.
[11]. A. Kumar Et Al., "A Network On Chip Architecture And Design Methodology," Ieee Computer Society Annual Symposium On Vlsi, 2002.
[12]. T. Bjerregaard And S. Mahadevan, "A Survey Of Research And Practices Of Network-On-Chip," Acm Computing Surveys, Vol. 38, No. 1, 2006.
[13]. M. Millberg Et Al., "Guaranteed Bandwidth Using Looped Containers In Temporally Disjoint Networks Within The Nostrum Network On Chip," Date, 2004.
[14]. P. Guerrier And A. Greiner, "A Generic Architecture For On-Chip Packet-Switched Interconnections," Date, 2000.
[15]. E. Rijpkema Et Al., "Trade-Offs In The Design Of A Router With Both Guaranteed And Best-Effort Services For Networks On Chip," Date, 2003.