International Journal of Electronic Devices and Networking

E-ISSN: 2708-4485
P-ISSN: 2708-4477
IJEDN 2023; 4(1): 52-57

www.electronicnetjournal.com
Received: 10-01-2023
Accepted: 13-02-2023

Abhinav Ark
Department of Electronics and Communication Engineering, Birla Institute of Technology, Patna, Bihar, India

Aakansha Kumari
Department of Electronics and Communication Engineering, Birla Institute of Technology, Patna, Bihar, India

Surabhi Dutta
Department of Electronics and Communication Engineering, Birla Institute of Technology, Patna, Bihar, India

Sweta Kumari
Department of Electronics and Communication Engineering, Birla Institute of Technology, Patna, Bihar, India

Megha Dadel
Department of Electronics and Communication Engineering, Birla Institute of Technology, Patna, Bihar, India

Correspondence Author;
Abhinav Ark
Department of Electronics and Communication Engineering, Birla Institute of Technology, Patna, Bihar, India

Design and simulation of asynchronous FIFO buffer using globally asynchronous and locally synchronous methodology

Abhinav Ark, Aakansha Kumari, Surabhi Dutta, Sweta Kumari and Megha Dadel

Abstract
Combining a faster system with a slower system might be difficult. To solve this issue, FIFO buffers are used. A FIFO buffer is called the First In, First Out buffer. The initial written data will be read from this non-random access memory type. Two systems that operate at distinct frequency levels are linked using a FIFO buffer. The high-frequency component is connected to the low-frequency component through a FIFO Buffer with a particular operating frequency to prevent data loss during transmission. This research paper uses globally asynchronous and locally synchronous method to build a FIFO Buffer and improve performance.

Keywords: First in first out (FIFO), asynchronous, globally asynchronous locally synchronous (GALS), behavioral modelling, structural modelling

1. Introduction
FIFOs have become integral to digital systems as the design has become more modular. The design's power, performance, and cost can be significantly impacted by the type of structured memory used [1]. Serial and parallel are the two types of architectural schemes and FIFO designs. The first FIFO generation was the serial FIFO (Such as shift register) that worked using a fall-through principle (Or pipeline). The parallel type of FIFOs are faster than serial FIFOs and were the second kind of the FIFOs used [2].
An asynchronous FIFO architecture uses two clock domains that are asynchronous to each other, wherein one clock domain is utilized to sequentially put data values into a FIFO buffer and the other clock domain is utilized to sequentially read the data from the same FIFO buffer. The throughput of the system is a critical productivity parameter.
It gives units of information processed at a given time. The limits of the analog physical medium being used, the components' processing capability, and end-user behaviour in a communication system can all impact throughput [3]. Throughput gauges the efficiency of hard drives, RAM, networking, and Internet connections. The purpose is to increase the throughput of the FIFO Queue Buffer and make it more efficient using globally asynchronous and locally synchronous methodology. A static memory component is utilized in two-pointer architecture for a FIFO buffer. The memory location and address of the data word are stored using pointers. This architecture does not perform data word shifting operations. Instead, read and write pointers are used twice.

2. FIFO Technology
Printed circuit boards (PCBs) that exchange data are present in every digital piece of equipment. When data reaches the printed circuit boards quickly or in large groups but is processed slowly or inconsistently, buffering, sometimes called interim storage, is always necessary. First In, First Out buffer is also known as FIFO. It is a type of non-random access memory where written data is read out first. Connecting two systems that run at various frequencies uses a FIFO buffer. A FIFO Buffer with a specific operating frequency connects the high-frequency and low-frequency components to avoid data loss during transmission. It can be implemented using the Muller C element or JTL in RFSQ [4].
One way of building FIFO buffers using a fall through principle using shift registers [3]. Control logic, storage, and read and write pointers comprise a hardware FIFO. Storage options include latches, flip-flops, static random-access memory (SRAM), etc.

For larger FIFOs, a dual port SRAM with reading and writing ports is employed [5]. A binary-coded address (write address and read address) is used to access the FIFO memory block. To determine the FIFO state (Full or empty), gray-coded address is generated from the binary- coded address and transferred to other clock domain [6]. Multi-bit signal change problem while latching into synchronizers is eliminated using gray-coded address transfer [7, 8]. This kind of architecture does not perform data word-shifting procedures. Instead, two read/write pointers are employed. Read pointers contain the memory address from which the data is read. The reading of the data progresses it. Write pointers advance and store the memory address where the data is being written after each writing operation. These pointers are used to generate status flag signals. Upon reset, the read and write pointer starts at zero. When a write pointer passes over a read pointer, the FIFO is said to be full; when a read pointer catches up to the write pointer, it is said to be empty [9, 10].

3. Globally Asynchronous Locally Synchronous

With the increase in design complexity and clock frequency of a digital circuit, using a synchronous clock has increased the power consumption and area on the chip. Another major problem that arises from using a synchronous clock is that it significantly contributes to the noise in analog circuit components such as phase-locked loops and signal converters. In place of synchronous blocks, if adequately designed, asynchronous blocks can be more robust, fast and power efficient. Asynchronous FIFO operation is based on a handshaking protocol between the sender (Active) and receiver (Passive) and is synchronized by request and acknowledge handshaking signals [11, 12]. Asynchronous FIFOs can tolerate variation better [13]. But this will add a large number of control overheads to the system. Thus, Globally Asynchronous Locally Synchronous methodology is used to overcome this challenge [14]. The idea behind GALS is to split a block into smaller sub-blocks. These sub-blocks are synchronous locally but are connected asynchronously. So instead of using a fixed period clock as used in globally synchronous systems, GALS systems have a locally generated clock whose period is specified for the local (Synchronous) block [15]. Thus, the block as a whole is Asynchronous globally. The locally synchronous blocks communicate using asynchronous handshaking [16, 17]. A common approach is to add an asynchronous wrapper, which provides an interface from the synchronous to the asynchronous environment (and vice versa), to every locally synchronous block. The asynchronous wrapper also controls asynchronous communication between locally synchronous blocks [18, 19, 20]. This methodology helps to increase the speed and decrease the power consumption, control overheads, electromagnetic induction, and die area.

4. Methodology

The working of the FIFO buffer is achieved using Verilog coding, a hardware description language. Verilog coding is easy to implement, and debugging can be easily done regardless of design size. The working methodology consists of four steps- coding, synthesis, implementation, and simulation.

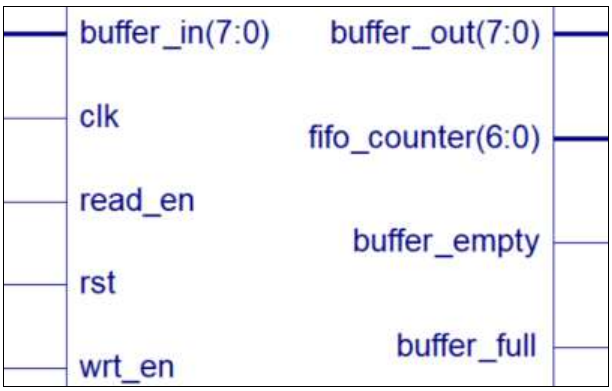

Fig 1: RTL Schematic of FIFO buffer

The coding portion involves writing the Verilog code according to the designed FIFO buffer's specifications. While coding, four types of pointers are mentioned: read pointer, write pointer, memory array, and status flag generator. The fig 1, is a general outline of the FIFO buffer showing the buffer input, clock, read enable , reset , write enable, buffer full status, buffer empty status, buffer output and the FIFO counter [21].

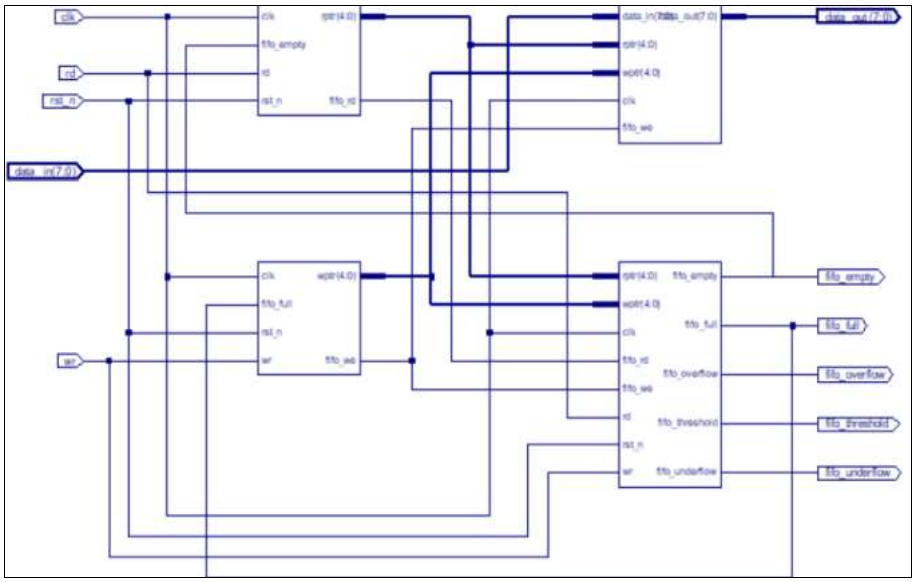

Fig 2: An in-depth view of the RTL Schematic of FIFO buffer

The synthesis process involves the generation of a netlist for every source file. Further, implementation divides the section into three parts - translate, map, place, and route. These three parts involve assembling several files into a single netlist, mapping I/O blocks, placing the design on the chip, and connecting the components. The fig 2, shows how the various components obtained after the synthesis process are implemented by mapping the design of the circuit and routing them. The summary of the design and report is generated after the implementation, followed by simulation. Errors, if any, are checked and corrected in the code during the simulation.

Procedural statements are used for behavioral modeling in the FIFO buffer. It uses a two-pointer method which is the read-and-write pointer.

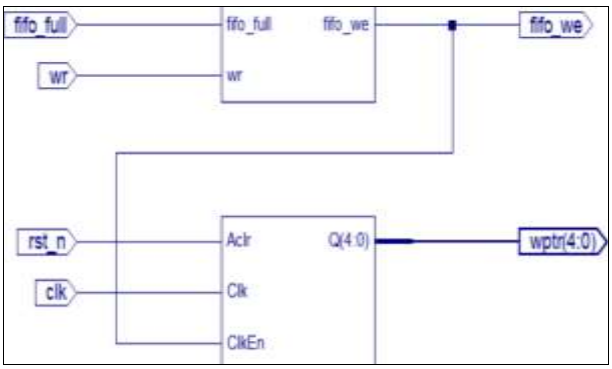

Fig 3: RTL Schematic of Write Pointer

For a write operation, the write pointer advances and stores the memory address where the data has to be written. There are high chances of overflow when the data is written while the FIFO is complete. Otherwise, the chances are low.

From fig 3, the RTL Schematic of the Write Pointer is seen.

The write operation is carried out if the FIFO buffer is not full. Once the buffer becomes full then the read enable is incremented and the write operation is stopped to prevent overflow of data. The write pointer always points to the location from where the write operation has to begin.

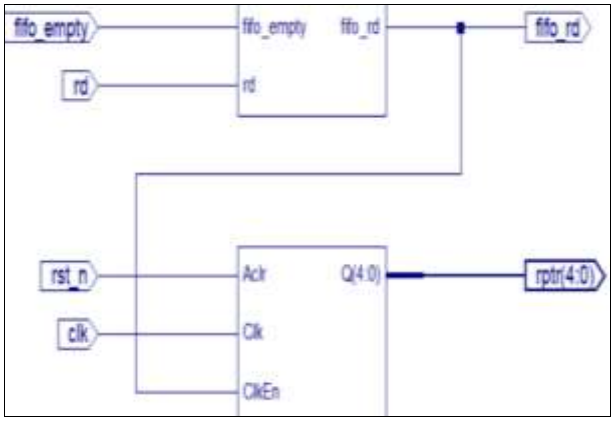

Fig 4: RTL Schematic of Read Pointer

The read pointer as shown in Fig 4, has the memory address from where the data has to be read. Upon reset, both the read and write pointers remain at zero. There are also high chances of underflow when data is read from FIFO while it is empty. Otherwise, the chances are low. The FIFO buffer also has a counter that increases by one for every write operation and decreases by one for every read operation.

The status flag generator indicates whether the read or write operation is being carried out. Fig 5 shows the RTL Schematic of the Status flag generator. The status flag becomes active when the buffer is full. This in turn enables the read to start the reading operation.

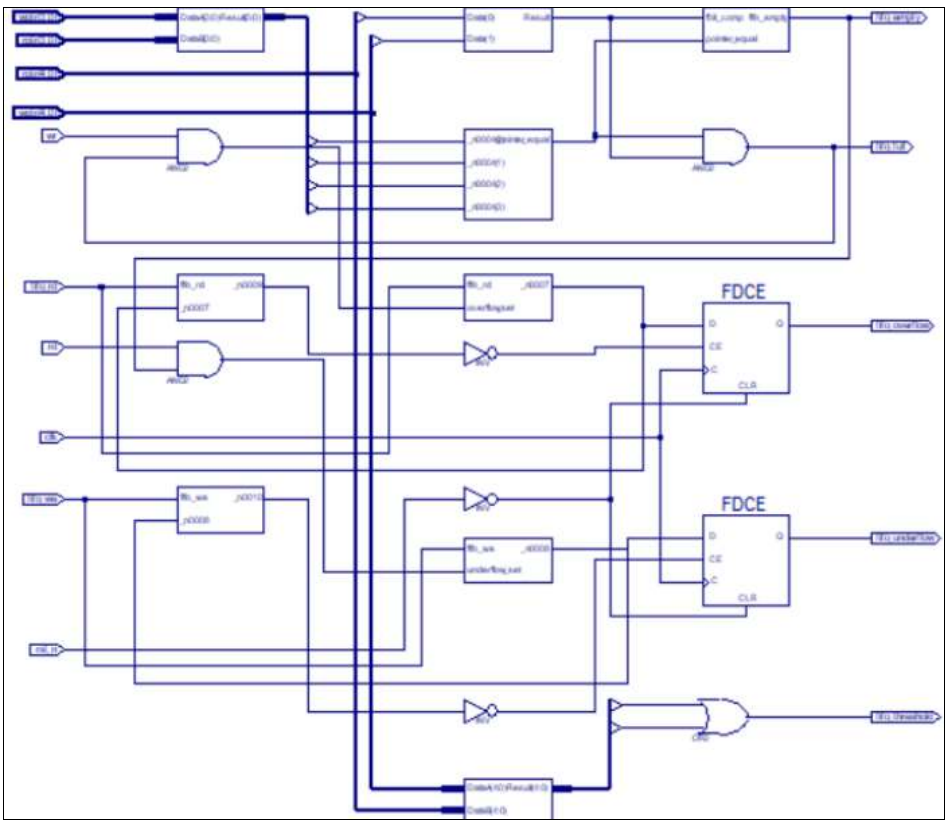

Fig 5: RTL Schematic of Status Flag Generator

Fig 6, gives an idea of the RTL Schematic of the memory array. The memory array stores the buffer input while the write operation is performed to prevent data loss. Once the write operation is done, the data is accessed from the memory array for a read operation using the read pointer.

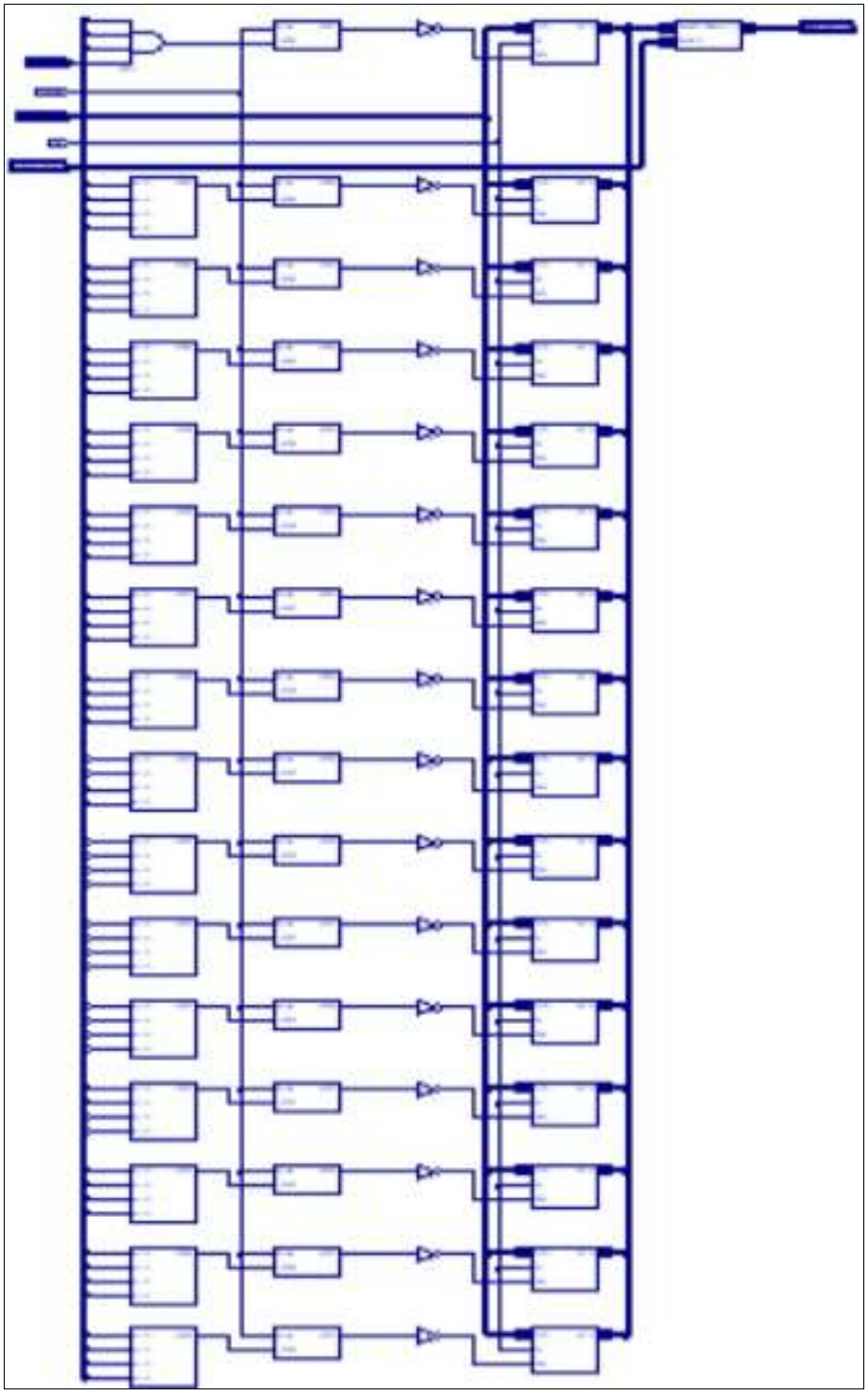

Fig 6: RTL Schematic of Memory Array

5. Design Simulation

Waveform, as shown in Fig 7, is obtained after the Testbench simulation is used to operate the FIFO buffer. Generally, the empty status flag is set to high whenever the FIFO buffer is empty and is out of data. Write enable is turned (logic 1) at the falling edge of the clock pulse. The value of the counter grows to one as the write pointer advances. After receiving 15 additional data inputs (buffer utilizes 16x8 memory array), the FIFO buffer becomes full, and the counter value reaches its limit making the full status signal rise to one.

Once the full status signal rises, no additional data can be written in the buffer to avoid data loss making the read enable switches too high for reading the data from the signal's head. When read enable is enabled the data is read at the FIFO buffer's output port and the counter drops to its minimum value of zero one by one. The empty flag is again set to high once all the data is read from the FIFO buffer [22].

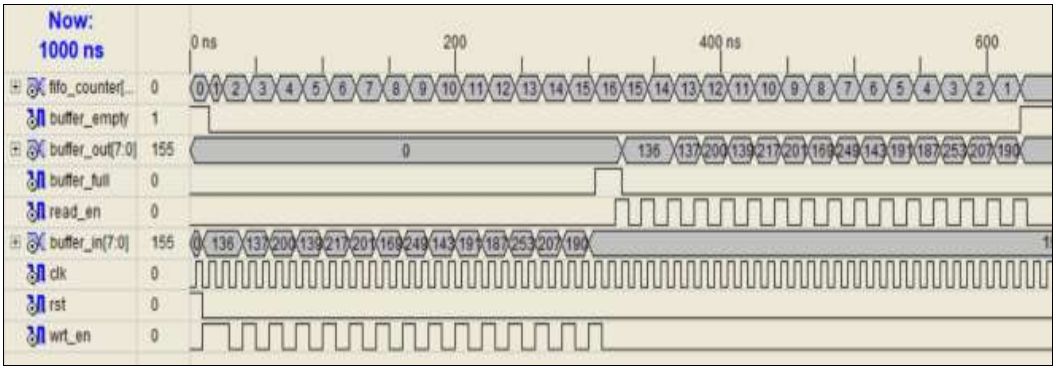

Fig 7: The waveform of testbench simulation

6. Conclusion

In this work, we implemented a FIFO buffer using the globally asynchronous and locally synchronous methodology for high throughput. The status flag generator, a memory component, a write pointer, a read pointer, and both the read and write enable. The read-and-write pointer indicates the memory location where data is being read or written. The FIFO counter's value goes up by one for every write action, and for every read operation, it goes down by one. When the read-and-write operation co-occurs, the value of the FIFO counter does not change. The queue's status signal lets you know if it's full or empty. This FIFO architecture can be used in a system operating on a high clock frequency to reduce power consumption and noise.

7. References

1. HoSuk Han Kenneth S. Stevens, Clocked and Asynchronous FIFO Characterization and Comparison, Electrical and the Computer Engineering University of Utah
2. Masoud Oveis-Gharana, Gul N Khan. Statically adaptive multi FIFO buffer architecture for network on chip, Electrical and Computer Engineering, Ryerson University, 350 Victoria St, Toronto, ON M5B 2K3, Canada, February; c2015.
3. Pragya Dour, Chhaya Kinkar, Throughput Improvement in Asynchronous FIFO Queue in Wired and Wireless Communication, Sagar Institute of Research Technology and Science RGPV University, Bhopal, India, 2016, 5(12). ISSN: 2278-0181 IJERTV5IS120136.
4. Herr QP, Bunyk P. Implementation and application of first-in-first-out buffers, TRW Space and Electronics Group, Redondo Beach, CA, USA, IEEE Transactions on Applied Superconductivity, 2003, 13(2).
5. Abdel-Hafeez S, Shatnawi M, Gordon-Ross A. A double data rate 8t-cell sram architecture for systems On-chip, IEEE 14th International Symposium on System-on-Chip; c2012.
6. Rahmani A, Liljeberg P, Plosila J, Tenhunen H. Design and implementation of reconfigurable FIFOs for Voltage/Frequency Island-based Networks-on-Chip. Microprocessors and Microsystems. 2013;37:432-445.
7. Hyoung-Kook Kim, Laung-Terng Wang, Yu-Liang Wu, Wen-Ben Jone. Testing of Synchronizers in Asynchronous FIFO
8. Gordon-Ross A, Abdel-hafeez S, Alsafrjalni MH. A one-cycle FIFO buffer for memory management units in Manycore System. IEEE Computer Society Annual Symposium on VLSI; c2019.
9. Fattah M, Manian A, Rahimi A, Mohammadi S. A High Throughput Low Power FIFO used for GALS NoC Buffers, IEEE Computer Society Annual Symposium on VLSI; c2010.
10. Chelcea T, Nowick SM. A low-latency FIFO for mixed-clock systems, Proceedings IEEE Computer Society Workshop on VLSI 2000. System Design for a System-on-Chip Era, Orlando, FL, USA; c2000. p. 119-126.
11. Saleh Abdel-Hafeez, Ann Gordon-Ross. Reconfigurable FIFO memory circuit for synchronous and asynchronous communication, Department of Computer Engineering, Jordan University of Science and Technology, Irbid, Jordan 2 Sabbatical at Department of Computer Engineering, College of Computer, Qassim University, Qassim, Buraydah, Saudi Arabia 3 Department of Electrical and Computer Engineering, University of Florida (UF), Gainesville, Florida, USA; c2020.
12. Abdel-Hafeez S, Quwaider MQ. A one-cycle asynchronous FIFO queue buffer circuit. 11th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan; c2020.
13. Taghi Adl SM, Mohammadi S. A high-performance dual clock elastic FIFO network Interface for GALS NoC. Microelectron J, Elsevier; c2018.
14. Keller B, Fojtik M, Khailany B. A Plausible Bisynchronous FIFO for GLAS systems. 21st IEEE International Symposium on Asynchronous Circuits and Systems, California; c2015. p. 1-8.
15. Dasgupta S, Yakovlev A. Comparative analysis of GALS clocking schemes, IET Comput. Digit. Techn. 2007;1(2):59-69.
16. Bormann DS, Cheung PYK. Asynchronous wrapper for heterogeneous systems, Proceedings International Conference on Computer Design VLSI in Computers and Processors, Austin, TX, USA; c1997. p. 307-314.
17. Mansi Jhamb RK, Sharma, Gupta AK. A Novel FIFO Design for Data Transfer in Mixed Timing Systems, Proceedings of International Journal of Electronics and Communication Engineering, 2014, 8(3).
18. Krstic M, Grass E, Gürkaynak FK, Vivet P. Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook. IEEE Design & Test of Computers. 2007;24(5):430-441.
19. Saleh Abdel-Hafeez1, A One-Cycle Asynchronous FIFO Queue Buffer Circuit, Muhannad Q. Quwaider2 1 Jordan University of Science and Technology; c2020.
20. Sheibanyrad A, Greiner A. Two efficient synchronous asynchronous converters well-suited for networks-on-

chip in GALS architectures, Integration. 2008;41(1):17-26,
21. HoSuk Han, Kenneth S Stevens. Clocked and Asynchronous FIFO Characterization and Comparison, Electrical and Computer Engineering, University of Utah, United States; c2009.
22. Panades M, Greiner A. Bi-Synchronous FIFO for Synchronous Circuit Well Suited for Network-on-Chip in GALS architectures, Proceedings of the First International Symposium on Networks-on-Chip (NOCS'07); c2007.