Implementation and Analysis on FIFO using FPGA

[1]A Roopa Sree, [2]E Bala Krishna, *Y. L. Ajay Kumar*[3]

[1]PG Scholar, [2]Assistant Professor, [3]Professor, Dept of ECE, Anantha Lakshmi institute of technology & sciences, Anantapuramu.

ABSTRACT

Several regions across India are currently experiencing significant energy shortages. This study investigates the implementation of an optimized First-In-First- Out (FIFO) mechanism using Field-Programmable Gate Array (FPGA) technology to enhance environmentally sustainable transmission systems. FIFO, which manages and processes items or data in the order they are received, mirrors the intuitive operation of real life queues and lines. The proposed design employs the Genesys board as the FPGA hardware, which offers high performance, Gigabit Ethernet connectivity, and design flexibility, making it suitable for highly complex applications. The integration of this optimized FIFO mechanism on the Genesys FPGA board demonstrates potential improvements in the efficiency and sustainability of energy transmission systems, addressing current energy shortfalls effectively.

Key Terms— Xilinx, FPGA, Energy effieciency, Viva do, First in First out (FIFO).

I. INTRODUCTION

FPGAs are semiconductor devices that are organised in the form of a matrix of configurable logic blocks (CLBs) that are linked together by means of programmable interconnects. FPGAs are widely used in various industries and applications, ranging from electronics and telecommunications to aerospace and automotive sectors [1]. At their core, FPGAs are made up of a network of programmable interconnects that allow the internal blocks to communicate with each other. These logic blocks can be configured to implement different logic functions, such as AND, OR, NOT, XOR, and more complex functions. Additionally, FPGA devices often include other specialized resources, such as memory blocks, digital signal processing (DSP) blocks, and circuitry for the management of clocks [2]. The FPGA is configured using hardware descriptive languages like VHDL and Verilog. This language is then compiled and synthesized into a configuration bit file that specifies how the logic blocks and interconnects should be set up to achieve the desired functionality. That bit file is loaded onto the FPGA device, effectively "programming" it to perform the specified logic operations [3]. One of the key advantages of FPGAs is their reconfigurability. Unlike traditional Application-Specific Integrated Circuits (ASICs) that are designed for a specific purpose and cannot be changed after manufacturing, FPGAs can be reprogrammed to adapt to different tasks or to fix errors or updates. This flexibility makes FPGA suitable for prototyping, testing, and even for use in products where design changes might be necessary over time[4]. FPGAs find applications in a wide range of fields, including: Digital Signal Processing: FPGAs can accelerate complex mathematical computations, making them useful in applications like image and audio processing. Communication Systems: FPGAs are used to implement communication protocols, encoding/decoding, and modulation/demodulation functions. Embedded Systems:

FPGAs can be integrated into embedded systems to provide hardware acceleration for specific tasks, reducing the load on the main processor. Aerospace and Défense: FPGAs are used in radar systems, satellite communication, avionics, and more due to their ability to handle real-time processing and adapt to changing requirements. Industrial Automation: FPGAs are used in industrial control systems for real-time monitoring and control of machinery and processes[1, 5]. Cryptocurrency Mining: FPGAs can be used to accelerate certain cryptographic calculations in mining operations. Prototyping: FPGAs are often used to prototype hardware designs before committing to manufacturing ASICs. Despite their versatility, FPGAs do have some limitations, such as lower performance compared to custom-designed ASICs for specific tasks and higher power consumption due to their general-purpose nature. However, advancements in FPGA technology continue to improve their performance, power efficiency, and capabilities. In computer science, a FIFO data structure is often referred to as a queue. In a queue, the first item added is the first one to be removed. For example, there is a long queue of people waiting to purchase tickets at the theatre. The individual who comes first will be given the opportunity to purchase a ticket before everyone else. In a similar manner, the item that is processed first in a computer programme is the one that was placed to the queue first.[6, 7]. In computer memory management, a FIFO page replacement algorithm is used to decide which page to remove from memory when space is needed for a new page. The idea is to replace the oldest page in memory, just like you would replace the oldest item in a queue. In business and accounting, FIFO is used to calculate the cost of goods sold (COGS) and the value of inventory. According to the FIFO principle, the items that were acquired or produced first are assumed to be sold or used first. This method can have an impact on tax liabilities and financial reporting. In networking, a FIFO buffer (First-In, First-Out buffer) is often used to manage the flow of data packets. Data packets are stored in the buffer in the order they arrive, and they are processed and sent out in the same order[8, 9]. FIFOs are also used in hardware design, especially in digital circuits. These FIFOs are used to manage data flow between different parts of a system, ensuring that data is handled in the order it was received.

II. EXISTING METHOD

The existing system for FIFO (First In, First Out) is a basic queueing mechanism commonly used in data storage and transfer applications. FIFO ensures that data is processed in the exact order it arrives, meaning that the first data element added to the queue is also the first one to be removed. This system operates through two main operations: **enqueue**, which adds data to the queue, and **dequeue**, which removes the data in the same order. FIFO is widely used in scenarios where maintaining the order of data is crucial, such as in communication systems, data buffering, and task scheduling. The mechanism works by maintaining two pointers—**head** and **tail**—where the head pointer indicates the position from which data will be removed, and the tail pointer marks where new data is added. FIFO is commonly implemented using various data structures like arrays or linked lists, but in hardware implementations, such as those on FPGAs (Field-Programmable Gate Arrays), the system is designed for high-speed, parallel processing. FPGA implementations of FIFO typically use **Block RAM (BRAM)** or other memory elements, with specific logic circuits for managing the enqueue and dequeue operations. These operations rely on read and write pointers to ensure the order of data flow, and the data elements are stored in registers or memory banks. The FPGA-based design allows the FIFO system to operate at high speeds with low latency, making it ideal for applications like data stream buffering in communication systems, packet switching in networks, or task scheduling in embedded systems. Furthermore, FPGA implementations offer flexibility in optimizing performance by adjusting the width of data paths, clock speeds, and memory configurations. However, the implementation of FIFO on FPGA requires careful consideration of resource utilization, such as the amount of available memory, logic blocks, and the overall design complexity. As FPGA technology allows for custom hardware designs, it provides the opportunity to fine-tune the FIFO system for specific use cases, enhancing its efficiency and performance compared to traditional software-based implementations. Through such an FPGA-based implementation, the FIFO system becomes highly scalable, with the potential for improved throughput, reduced latency, and better integration into larger hardware systems.

III. PROPOSED METHOD

Being a hardware solution, the FPGA-based FIFO system has a number of benefits over conventional software-based FIFO implementations, especially with regard to speed, flexibility, and dependability. With the help of a hardware platform called FPGA (Field-Programmable Gate Array), users can create unique circuits for particular applications, enabling high-performance execution. The FIFO logic in your suggested system is implemented by the FPGA, guaranteeing that data is handled effectively and processed in the precise order that it comes, without needless delays. Using FPGA for FIFO has the primary benefit of parallel processing. The architecture of FPGA allows it to do several operations at once, in contrast to software systems that usually process data in a sequential fashion. FIFO processes are accelerated by this parallelism, especially in applications that need real-time data management, including packet switching or data buffering. Additionally, FPGA has very low latency when managing data flow in high-speed applications including real-time embedded systems, communication systems, and video processing. Future Potential and Optimization: By utilizing newer features like larger memory blocks and high-speed interfaces, the suggested system can be further optimized as FPGA technology advances. More sophisticated processing methods, including caching to speed up enqueue and dequeue processes or dynamic memory allocation to maximize data storage and retrieval, can be incorporated into the system in later iterations. Furthermore, the FPGA FIFO system can be enhanced to manage even larger data volumes while preserving its effectiveness and low-latency performance when more complicated applications emerge.

TABLE 1

Results for Proposed Logic

Circuit	Static Power(nW)	Dynamic Power(fW)	Delay(ns)
NAND gate	20.1	0.49	4.43
NOR gate	11.6	1.30	4.42
Full Adder	10.2	0.82	3.87

Power

FIGURE 1. Dynamic Power for Domino and Proposed Logic

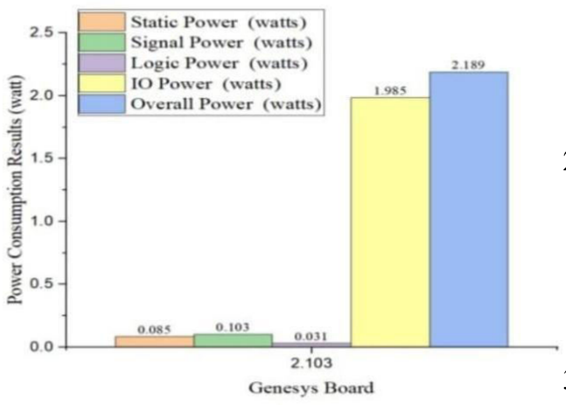

Fig. power consumption results

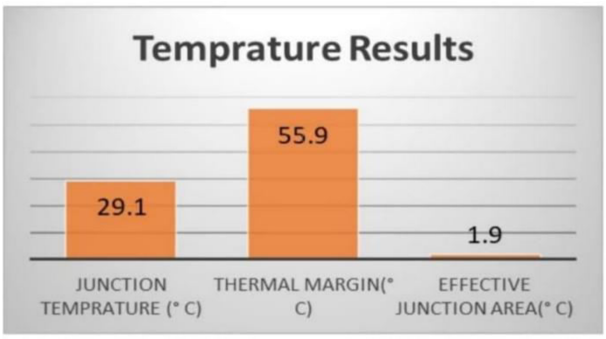

Fig: Temperature results

IV. RESULTS

Title and Author Details

Title must be in 12 pt Times New Roman font. Author name must be in 11 pt Regular font. Author affiliation must be in 10 pt Italic. Email address must be in 9 pt Courier Regular font.

Section Headings

No more than 3 levels of headings should be used. All headings must be in 10pt font. Every word in a heading must be capitalized except for short minor words as listed in Section III-B.

1) Level-1 Heading: A level-1 heading must be in Small Caps, cantered and numbered using uppercase Roman numerals. For example, see heading "III. Page Style" of this document. The two level-1 headings which must not be numbered are "Acknowledgment" and "References".

2) Level-2 Heading: A level-2 heading must be in Italic, left-justified and numbered using an uppercase alphabetic letter followed by a period. For example, see heading "C. Section Headings" above.

3) Level-3 Heading: A level-3 heading must be indented, in Italic and numbered with an Arabic numeral followed by a right parenthesis. The level-3 heading must end with a colon. The body of the level-3 section immediately follows the level-3 heading in the same paragraph. For example, this paragraph begins with a level-3 heading.

A. Figures and Tables

Place figures and tables at the places where they needed. All tables should be in Classic 1 format with borders to heading and subheading columns. Large figures and tables may span across both columns. To do so select text above one column table and convert it in two column and then select text below one column table and convert it into two column. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence. We suggest that you use border for graphic (ideally 300 dpi), with all fonts embedded) and try to reduce the size of figure to be adjust in one column. Figure and Table Labels: Use 8 point Times New Roman for Figure and Table labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader.

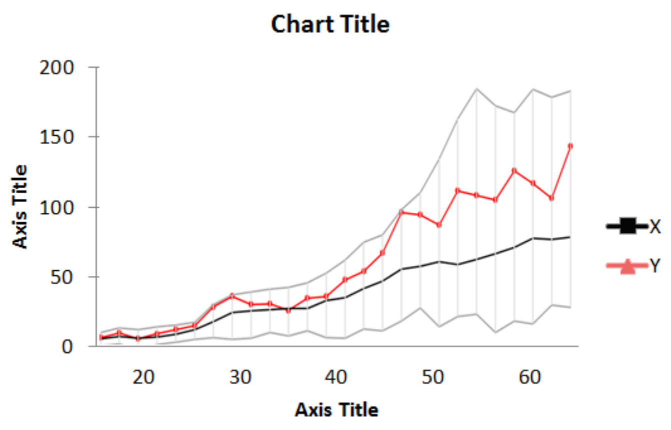

Figure 1: A sample line graph using color's which contrast well both on screen and on a black-and-white hardcopy

B. Page Numbers, Headers and Footers

Page numbers, headers and footers must not be used.

C. Links and Bookmarks

All hypertext links and section bookmarks will be removed from papers during the processing of papers for publication. If you need to refer to an Internet email address or URL in your paper, you must type out the address or URL fully in Regular font.

V. CONCLUSION

Using a Genesys board, FIFO is built in Vivado for the purpose of this research. The Genesys board's reliable and flexible environment makes it a good choice for developing FPGA-based systems, making it a viable alternative for a wide range of tasks BUFG, LUT, FF, and IO is right. Individual % breakdown: 1%, 0.008%, 0.002%, and 3.13% This chip needs 2.189 W (total).

VI. REFERENCES

[1] S. Gandhare and B. Karthikeyan, "Survey on FPGA architecture and recent applications," in 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (VI Tecon), 2019: IEEE, pp. 1-4.

[2] B. Zhang, R. Kannan, and V. Prasanna, "Boostgcn: A frame work for optimizing gcn inference on fpga," in 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2021: IEEE, pp. 29-39.

[3] R. J. Hayne, "Translating the Instructional Processor from VHDL to Verilog," in 2018 ASEE Annual Conference & Exposition, 2018.

[4] T. Wang, C. Wang, X. Zhou, and H. Chen, "An overview of FPGA based deep learning accelerators: challenges and opportunities," in 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/ Smart City/ DSS), 2019: IEEE, pp. 1674-1681.

[5] D. Koch, N. Dao, B. Healy, J. Yu, and A. Attwood, "FABulous: An embedded FPGA framework," in The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2021, pp. 45-56.

[6] G.-M. Sung, L.-F. Tung, H.-K. Wang, and J.-H. Lin, "USB transceiver with a serial interface engine and FIFO queue for efficient FPGA-to-FPGA communication," IEEE Access, vol. 8,

[7] V. Yadav, S. Kashyap, R. Pandey, and J. Madan, "Impact of Doping Variation on the Performance of Sb 2 S 3 based Solar Cell using Device Simulations," in 2023 IEEE Devices for Integrated Circuit (DevIC), 2023: IEEE, pp. 52-55.

[8] S. Parker et al., "Impact of FIFO work arrangements on the mental health and wellbeing of FIFO workers," 2018.

[9] A. Finkel and M. Praveen, "Verification of flat FIFO systems," Logical Methods in Computer Science, vol. 16, 2020.

[10] A. C. Sembiring, J. Tampubolon, D. Sitanggang, and M. Turnip, "Improvement of inventory system using first in first out (FIFO) method," in Journal of Physics: Conference Series, 2019, vol. 1361, no. 1: IOP Publishing, p. 012070.

[11] S. Rawat, S. Kashyap, J. Madan, and R. Pandey, "Numerical Simulations of FASnI 3 based Solar Cell with the Variation of Absorber Layer Thickness," in 2023 2nd Edition of IEEE Delhi Section Flagship Conference (DELCON), 2023: IEEE, pp. 1-4.

[12] A. Singla, A. Kaur, and B. Pandey, "LVCMOS based energy efficient solar charge sensor design on FPGA," in 2014 IEEE 6th India International Conference on Power Electronics (IICPE), 2014: IEEE, pp. 1-5.

[13] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex™-II FPGA family," in Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field programmable gate arrays, 2002, pp. 157-164.

[14] A. D. Brant, "Coarse and fine grain programmable overlay architectures for FPGAs," University of British Columbia, 2013.

[15] F. Siddiqui et al., "FPGA-based processor acceleration for image processing applications," Journal of Imaging, vol. 5, no. 1, p. 16, 2019.

[16] S. Das, U. Basu, R. Das, S. Saha, and A. Basu, "FPGA Implementation of Asynchronous FIFO," in Proceedings of International Conference on Industrial Instrumentation and Control: ICI2C 2021, 2022: Springer, pp. 399-407.

[17] M. N. Emas, A. Baylis, and G. Stitt, "High-frequency absorption fifo pipelining for stratix 10 hyperflex," in 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2018: IEEE, pp. 97-100.

[18] S. Attia and V. Betz, "Toward Software-Like Debugging for FPGAs via Checkpointing and Transaction-Based Co-Simulation," ACM Transactions on Reconfigurable Technology and Systems vol. 16, no. 2, pp. 1-24, 2023.