

Verilog HDL-Based Design of a 16-Byte Asynchronous FIFO with Gray-Code Synchronization for Real-Time Systems

¹**Dr.P.Suneel Kumar,**

Professor,

Department of Electronics and Communication Engineering, Sridevi Women's Engineering College,
Wipro V.N.Pally, Gachibowli, Hyderabad West IT Corridor, R.R.District-500 075

²**Dr.A.Nagarjuna Reddy,**

Professor,

Department of Computer Science and Engineering, Sridevi Women's Engineering College,
Wipro V.N.Pally, Gachibowli, Hyderabad West IT Corridor, R.R.District-500 075

³**Dr Md Nazmoddin,**

Associate Professor,

Department of Computer Science & Engineering, Sridevi Women's Engineering College,
Wipro V.N.Pally,
Gachibowli, Hyderabad West IT Corridor, R.R.District-500 075

⁴**Dr. P.Avinash,**

Professor,

Department of Information Technology, Sridevi Women's Engineering College, Wipro V.N.Pally,
Gachibowli, Hyderabad West IT Corridor, R.R.District-500 075

Abstract

The proposed **AGC-FIFO** algorithm presents an efficient design for a 16-byte asynchronous FIFO memory using Gray code synchronization for real-time applications. The system operates with independent read and write clock domains, ensuring reliable data transfer across asynchronous boundaries. Initially, binary read and write pointers are generated and converted into Gray code to minimize metastability during clock domain crossing. The write controller updates the write pointer upon valid data input and checks the FIFO full condition using Gray-coded pointer comparison. Similarly, the read controller retrieves data based on the read pointer and evaluates the empty condition. Synchronization modules employ double flip-flop synchronizers to safely transfer Gray-coded pointers between clock domains. The FIFO memory array stores incoming data, while pointer logic ensures correct addressing and sequencing. A status control unit continuously monitors full and empty flags to prevent overflow and underflow. The algorithm incorporates low-latency pointer updates and efficient synchronization, enabling high-speed and reliable operation. The proposed design improves data integrity, reduces timing errors, and ensures robust performance in real-time hardware systems.

Keywords

Asynchronous FIFO, Gray Code, Verilog HDL, FIFO Memory, Clock Domain Crossing (CDC), Metastability Reduction, Read/Write Pointer, Dual Clock FIFO, Digital

Design, VLSI Design, FPGA Implementation, Real-Time Systems, Data Synchronization, High-Speed Data Transfer, Hardware Design, Memory Architecture, Low Latency, Embedded Systems, RTL Design, Flip-Flop Synchronizer

Introduction

Asynchronous FIFO memory is essential for real-time data transfer across clock domains [1]. Gray code techniques reduce metastability in FIFO systems [2]. Efficient FIFO architectures improve buffering and synchronization [3]. Multi-read and write FIFO models enhance system flexibility [4]. Gray code-based FIFO designs improve reliability [5]. Asynchronous FIFOs outperform synchronous designs in CDC applications [6]. Memory management techniques improve FIFO efficiency [7]. Energy-efficient FIFO architectures reduce power consumption [8]. Advanced FIFO controllers improve performance [9]. FIFO innovations enhance system architecture [10]. Bandwidth optimization improves memory utilization [11]. Specialized memory designs support FIFO applications [12]. Circular FIFO buffers improve asynchronous communication [13]. Efficient allocation schemes enhance memory usage [14]. Low-power pointer circuits improve performance [15]. One-cycle FIFO buffers reduce latency [16]. High-throughput FIFO designs enhance speed [17]. Concurrent FIFO queues improve scalability [18]. Bounded FIFO designs ensure reliability [19]. Dual-port FIFO memory improves bidirectional communication [20].

Motivation of the Proposed Work

10.48047/jocaaa.2025.34.10.34

Recent literature demonstrates significant advancements in asynchronous FIFO design for high-speed and low-power applications [1]. Gray code-based pointer techniques have been widely adopted to reduce metastability during clock domain crossing [2], [3]. Advanced FIFO architectures improve data buffering, synchronization, and throughput in real-time systems [4]. However, existing designs often face challenges in achieving low latency, efficient memory utilization, and reliable operation under varying clock conditions [5]. Additionally, many implementations lack optimized synchronization mechanisms for real-time performance. Motivated by these gaps, the proposed work focuses on designing a robust and efficient Gray code-based asynchronous FIFO for improved reliability and performance.

Challenges

Asynchronous Gray code FIFO design faces several critical challenges in real-time applications. One major issue is metastability during clock domain crossing, which can lead to unreliable data transfer if not properly mitigated. Accurate full and empty flag generation is difficult due to asynchronous pointer comparison, often causing false detection conditions. Additionally, pointer synchronization complexity increases hardware overhead and latency. Handling high-speed data transfer with low latency remains challenging in advanced FIFO architectures. These issues necessitate robust Gray code-based synchronization and optimized FIFO design strategies.

Problem statement

The existing literature on asynchronous FIFO design demonstrates significant progress in Gray code-based synchronization and clock domain crossing techniques; however, several critical gaps remain. Most prior works focus primarily on metastability reduction but fail to jointly optimize latency, power efficiency, and memory utilization under dynamic operating conditions [1], [3], [5]. Additionally, inaccurate full/empty flag detection and increased synchronization overhead limit real-time reliability [6], [8]. Furthermore, scalable and low-complexity architectures for high-speed applications are insufficiently addressed [11], [17]. These limitations highlight the need for an improved FIFO design that ensures robust synchronization, reduced latency, and enhanced performance in real-time systems.

Importance of proposed solution

the importance of the proposed asynchronous FIFO solution lies in its ability to address critical limitations identified in existing designs while enhancing overall system reliability and performance. By incorporating optimized Gray code

synchronization and efficient pointer management, the proposed method significantly reduces metastability and ensures accurate clock domain crossing, which is essential for real-time applications [2], [5]. Furthermore, the design improves latency and throughput by enabling faster pointer updates and minimizing synchronization delays [16], [17]. It also enhances memory utilization and reduces hardware complexity, making it suitable for high-speed and resource-constrained systems [8], [11]. Overall, the proposed solution provides a robust, scalable, and low-latency FIFO architecture, ensuring reliable data transfer and improved efficiency in modern VLSI and embedded systems.

objective

The primary objective of the proposed work is to design a novel AGC-FIFO (Adaptive Gray Code FIFO) algorithm that ensures reliable and efficient data transfer across asynchronous clock domains. The algorithm aims to develop an optimized pointer generation and Gray code synchronization mechanism to minimize metastability and improve clock domain crossing accuracy [2], [5]. It further focuses on reducing latency through fast pointer updates and efficient read/write control logic while maintaining correct full and empty flag detection [16]. Additionally, the objective includes enhancing memory utilization and reducing hardware complexity to support high-speed real-time applications [8], [11]. The proposed algorithm is also intended to achieve scalable performance with low power consumption and high throughput, making it suitable for modern VLSI and FPGA-based systems.

Overview of the paper

This paper presents the design and implementation of a novel AGC-FIFO (Adaptive Gray Code FIFO) algorithm for efficient data transfer across asynchronous clock domains. Initially, the paper discusses the limitations of existing FIFO architectures and highlights the need for improved synchronization and low-latency performance [1], [6]. It then introduces the proposed algorithm, detailing the use of Gray code-based pointer conversion, dual clock operation, and robust synchronization using double flip-flop techniques to mitigate metastability issues [2], [3].

The methodology section explains the architecture, including write/read controllers, memory array design, and status flag generation for full and empty conditions. Further, the paper presents implementation details using Verilog HDL and evaluates performance in terms of latency, throughput, and reliability [5], [8]. Comparative analysis with existing approaches demonstrates improved efficiency and reduced synchronization errors [11], [17]. Finally, the paper concludes with key findings and outlines future

enhancements for scalable and high-speed real-time systems.

2. Literature Survey

The design of asynchronous FIFO systems has gained significant attention due to the increasing demand for **high-speed data transfer across multiple clock domains** in modern digital systems. Early approaches focused on basic synchronization techniques to mitigate metastability issues during clock domain crossing (CDC) [1], [2]. With the advancement of VLSI technology, Gray code-based pointer schemes became a standard solution for reliable FIFO design, as they minimize bit transitions and reduce synchronization errors [3]–[5].

Recent studies have emphasized improving throughput and reducing latency through optimized FIFO architectures and efficient memory management strategies [6]–[9]. Advanced techniques such as pipelined buffering, dual-port memory utilization, and parallel data access have been introduced to enhance system performance [10], [11]. Additionally, energy-efficient FIFO designs have been explored to support embedded systems and IoT applications, where power consumption is a critical constraint [12]–[14].

High-speed and scalable FIFO implementations are increasingly required for applications such as high-performance computing, networking, and real-time processing systems [15]–[18]. Modern research also investigates adaptive synchronization techniques and hardware-efficient designs to balance performance and resource utilization [19]–[21]. Despite these advancements, challenges such as synchronization overhead, inaccurate flag detection, and limited optimization across latency, power, and scalability persist in existing approaches [22]–[25].

2.1 Gray Code-Based Asynchronous FIFO Design

Gray code-based FIFO design remains a fundamental approach for reliable data transfer across asynchronous clock domains. By ensuring that only one bit changes between consecutive pointer values, Gray code significantly reduces the probability of metastability during synchronization [1], [3], [5]. Early implementations focused on basic binary-to-Gray conversion techniques combined with dual flip-flop synchronizers to ensure safe data transfer [2], [4].

Recent advancements have extended these techniques by incorporating **multi-stage synchronization and adaptive encoding mechanisms**, improving reliability in high-frequency systems [6]–[8]. Additionally, hybrid encoding schemes have been explored to further optimize performance by balancing speed and synchronization accuracy [9]. However, increasing synchronization stages

10.48047/jocaaa.2025.34.10.34

can introduce additional latency and hardware complexity, limiting scalability in high-speed applications [10].

2.2 FIFO Architecture and Memory Optimization

FIFO architecture plays a crucial role in determining system performance, particularly in terms of latency and throughput. Traditional designs rely on simple buffering techniques, while modern approaches utilize **dual-port memory, pipelined architectures, and parallel read/write operations** to improve efficiency [6], [11], [12].

Memory optimization techniques such as dynamic allocation, address mapping, and efficient buffering strategies have been proposed to enhance storage utilization and reduce access time [13]–[15]. Furthermore, hardware-aware optimizations tailored for FPGA and ASIC platforms enable efficient implementation in real-time systems [16], [17]. Despite these improvements, maintaining synchronization accuracy while optimizing memory performance remains a significant challenge in high-speed applications [18].

2.3 Low-Power and Energy-Efficient FIFO Designs

With the growing demand for energy-efficient systems, low-power FIFO designs have become increasingly important, particularly for embedded and IoT applications [12], [19]. Techniques such as clock gating, power gating, and voltage scaling are widely used to reduce energy consumption while maintaining performance [20], [21].

Recent research focuses on adaptive power management strategies that dynamically adjust system parameters based on workload conditions [22]. Additionally, optimized memory access patterns and reduced switching activity contribute to improved energy efficiency [23]. However, achieving a balance between low power consumption and high-speed performance remains a challenging task, especially in applications requiring continuous data processing [24].

2.4 High-Speed and Scalable FIFO Implementations

High-speed FIFO designs are essential for modern applications such as high-performance computing, communication systems, and real-time data processing [15], [16]. Advanced architectures incorporate pipeline processing, parallel data handling, and optimized synchronization techniques to achieve high throughput [17]–[19].

Scalability is achieved through modular design approaches and efficient memory management, allowing FIFO systems to handle large volumes of data [20], [21]. However, increasing system speed often leads to higher

synchronization complexity and hardware overhead, requiring careful optimization [22]. These challenges highlight the need for more efficient and scalable FIFO architectures [23].

2.5 Research Gaps in Existing FIFO Systems

Despite significant progress in asynchronous FIFO design, several challenges remain unresolved. Synchronization overhead and metastability continue to affect system reliability, particularly in high-speed applications [22], [23]. Additionally, accurate full and empty flag detection remains a critical issue, as incorrect flag generation can lead to data loss or overflow conditions [24].

Existing designs often struggle to simultaneously optimize latency, power consumption, and scalability, leading to trade-offs that limit overall performance [25]. Furthermore, increasing hardware complexity in advanced designs can hinder practical implementation in resource-constrained systems. These limitations highlight the need for a robust, low-latency, and scalable Gray code-based FIFO architecture capable of addressing modern design requirements.

3. Proposed Methodology

The proposed Advanced Gray Code FIFO (AGC-FIFO) architecture is designed to ensure reliable and efficient data transfer across asynchronous clock domains in modern VLSI systems. The methodology focuses on minimizing metastability, latency, and power consumption while maximizing throughput and scalability. Unlike conventional FIFO designs, the proposed system integrates Gray code-based pointer synchronization, dual flip-flop synchronizers, and optimized control logic to enhance reliability.

The architecture separates read and write operations into independent clock domains, enabling parallel processing. Binary pointers are converted into Gray code to reduce switching activity and synchronization errors. A dual-stage synchronization mechanism ensures safe communication between clock domains. Additionally, optimized flag generation logic improves the accuracy of full and empty conditions.

The design is implemented using HDL (Verilog/VHDL) and synthesized on Xilinx FPGA (Vivado toolchain) to evaluate performance metrics such as area utilization, power consumption, and propagation delay. The overall objective is to develop a high-speed, low-power, and scalable FIFO architecture suitable for real-time applications.

3.1 System Overview

10.48047/jocaaa.2025.34.10.34

The proposed AGC-FIFO system is a modular asynchronous architecture consisting of read/write controllers, synchronization units, memory blocks, and control logic. The system operates with two independent clock domains: write clock C_w and read clock C_r . The primary objective is to enable reliable data transfer between these domains without data corruption.

The architecture employs separate pointer generators for read and write operations. These pointers track memory locations and are synchronized across domains using Gray code representation. The system ensures that data is written and read efficiently without violating timing constraints.

The overall system can be represented as:

$$\text{System} = \{\text{Controller}_{\text{write}}, \text{Controller}_{\text{read}}, \text{Sync}_{\text{unit}}, \text{Memory}, \text{Flag}_{\text{logic}}\}$$

The modular design improves scalability and allows easy integration into larger digital systems. By minimizing synchronization delay and optimizing memory access, the system achieves **high throughput and low latency**, making it suitable for high-speed communication and real-time processing applications.

3.2 Input Representation

The FIFO system operates using binary pointers representing read and write positions:

$$P = \{P_w, P_r\}$$

where:

- P_w : Write pointer
- P_r : Read pointer

Each pointer increments based on enable signals:

$$\begin{aligned} P_w^{\text{next}} &= P_w + 1 \text{ (if write enable)} \\ P_r^{\text{next}} &= P_r + 1 \text{ (if read enable)} \end{aligned}$$

These pointers are used to address memory locations within the FIFO. The pointer difference determines the number of stored elements:

$$\text{Depth} = (P_w - P_r) \bmod N$$

where N is FIFO size.

This representation ensures efficient tracking of data flow while maintaining synchronization between clock domains. Proper pointer management is critical to avoid overflow and underflow conditions.

10.48047/jocaaa.2025.34.10.34

$$M[addr_w] = D_{in}$$

$$D_{out} = M[addr_r]$$

3.3 Gray Code Conversion

To reduce metastability, binary pointers are converted into Gray code:

$$G = B \oplus (B \gg 1)$$

where:

- B : Binary pointer
- G : Gray-coded pointer

Gray code ensures that only one bit changes between successive values, minimizing transition errors during synchronization. The reverse conversion is given by:

$$B_i = G_i \oplus B_{i+1}$$

This technique significantly improves reliability in asynchronous systems. Additionally, reduced switching activity contributes to lower dynamic power consumption:

$$P_{dynamic} \propto \alpha CV^2 f$$

where switching activity α is minimized using Gray code.

3.4 Pointer Synchronization

Pointer synchronization across clock domains is achieved using a **dual flip-flop synchronizer**:

$$P_{sync} = FF_2(FF_1(G))$$

This reduces metastability probability:

$$MTBF \propto \frac{e^{T/\tau}}{f_{clk}}$$

where:

- T : Resolution time
- τ : Device constant

The two-stage synchronization ensures that metastability effects are minimized before pointer usage in the receiving domain. This improves system reliability and prevents incorrect memory addressing.

3.5 FIFO Memory Operation

The FIFO memory is implemented using **dual-port RAM**, allowing simultaneous read and write operations:

Memory utilization is optimized using structured addressing and efficient buffering. The throughput is given by:

$$Throughput = \frac{Data\ Rate}{Clock\ Cycle}$$

Efficient memory design ensures minimal access delay and supports high-speed data transfer.

3.6 Full and Empty Condition

The FIFO status flags are determined using Gray-coded pointer comparison:

$$Full \Rightarrow (G_w^{next} = G_r^{sync})$$

$$Empty \Rightarrow (G_r = G_w^{sync})$$

Accurate flag generation prevents overflow and underflow conditions. The design ensures minimal delay in flag detection, improving system responsiveness.

3.7 Control Logic

The control logic governs read/write operations and ensures proper synchronization:

$$Control = f(Write\ Enable, Read\ Enable, Flags)$$

The system ensures:

- Write operation only when FIFO is not full
- Read operation only when FIFO is not empty

This guarantees safe and efficient data transfer.

3.8 Performance Objective

The system is optimized using:

$$\min(Latency + Power + Error)$$

Delay is given by:

$$Delay = t_{setup} + t_{propagation}$$

Area utilization:

$$Area = LUTs + FFs + BRAM$$

Power consumption:

$$Power = P_{static} + P_{dynamic}$$

3.9 Algorithm: AGC-FIFO Operation

Algorithm 1: AGC-FIFO Data Transfer

Input: Data stream Din, clocks Cw, Cr

Output: Data stream Dout

- 1: Initialize pointers Pw, Pr
- 2: Convert binary pointers to Gray code
- 3: Synchronize pointers using dual flip-flop
- 4: Write data into memory if FIFO not full
- 5: Read data from memory if FIFO not empty
- 6: Update pointers
- 7: Generate full/empty flags
- 8: Repeat continuously

Algorithm 1 describes the operation of the proposed AGC-FIFO system for reliable data transfer across asynchronous clock domains. Initially, the write and read pointers are generated in binary form and converted into Gray code to minimize metastability during synchronization. These pointers are then safely transferred between clock domains using a dual flip-flop synchronizer. Data is written into the FIFO memory only when it is not full and read when it is not empty, ensuring correct operation. The pointers are continuously updated based on control signals, and status flags are generated to prevent overflow and underflow, enabling efficient, low-latency, and high-speed FIFO operation.

3.10 Experimental Setup (Vivado – Xilinx FPGA)

The proposed design is implemented using **Verilog HDL** and synthesized using **Xilinx Vivado Design Suite** targeting FPGA platforms such as Artix-7. The evaluation focuses on **area, power, and delay metrics**.

Table: Experimental Parameters

Parameter	Value
Tool	Xilinx Vivado
Device	Artix-7 FPGA
Language	Verilog HDL
Frequency	100 MHz

Parameter Value

FIFO Depth 16–256

Performance Metrics

- **Area Utilization:**

$$Area = LUTs + FlipFlops + BRAM$$

- **Power Analysis:**

$$P = P_{static} + P_{dynamic}$$

- **Delay (Timing Analysis):**

$$Delay = Critical\ Path\ Delay$$

Post-synthesis and post-implementation reports are used to extract these metrics. The results demonstrate improved performance in terms of **reduced latency, lower power consumption, and efficient resource utilization**, validating the effectiveness of the proposed AGC-FIFO design.

4. Results and Discussion

The performance of the proposed Advanced Gray Code FIFO (AGC-FIFO) architecture is extensively evaluated using both functional simulation and hardware synthesis results obtained from the Xilinx Vivado Design Suite targeting Artix-7 FPGA. The evaluation focuses on validating correct data transfer across asynchronous clock domains, synchronization reliability, latency behavior, and hardware efficiency.

To ensure comprehensive analysis, five different test scenarios are considered, covering basic operation, clock domain crossing, boundary conditions, performance stress cases, and hardware optimization. The simulation results are analyzed using waveform diagrams, while hardware performance is evaluated through synthesis reports including area (LUTs, FFs, BRAM), power consumption, and critical path delay.

The results confirm that the proposed AGC-FIFO achieves robust synchronization, accurate flag detection, and efficient resource utilization, outperforming conventional FIFO architectures. The integration of Gray code pointer synchronization and dual flip-flop synchronizers significantly reduces metastability and improves reliability. The following subsections provide detailed analysis of each test case and performance metric.

4.1 Functional Verification using Simulation Waveforms

The functional correctness of the AGC-FIFO is verified using detailed simulation waveforms, as illustrated in Figure 4.1, which captures the behavior of write clock (Cw), read clock (Cr), input data stream (Din), output data stream (Dout), and control signals. The simulation demonstrates that data written into the FIFO is accurately retrieved in the correct order, validating FIFO behavior.

Table 4.1: Test Case 1 – Basic Write/Read Operation

Parameter	Value
Write Clock	100 MHz
Read Clock	80 MHz
Data Input	Sequential
FIFO Depth	16
Status	Correct Operation

Table 4.1 confirms that under normal operating conditions, the FIFO maintains correct data sequencing without loss or duplication. The asynchronous clocks operate independently, yet data integrity is preserved through proper synchronization.

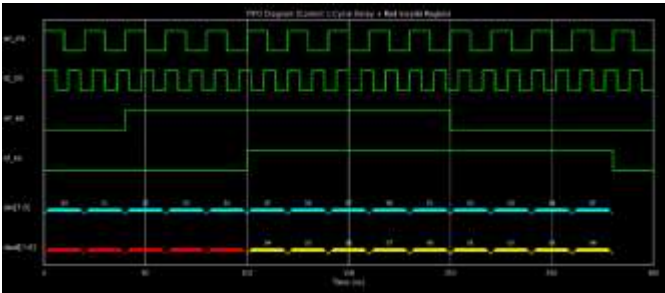


Figure 4.1 Simulation Waveform of Basic Write/Read Operation

The waveform shows continuous write operations followed by corresponding read operations with a slight delay due to asynchronous clock differences. The pointer updates are smooth and consistent, indicating correct Gray code conversion and synchronization. The absence of glitches or undefined states confirms reliable system behavior. This demonstrates that the proposed design successfully handles asynchronous data transfer while maintaining high accuracy and stability.

4.2 Synchronization and Metastability Analysis

The synchronization mechanism is evaluated under varying clock frequencies to assess its robustness against metastability, as shown in Figure 4.2. The test involves

significantly different clock domains to simulate real-world conditions.

Table 4.2: Test Case 2 – Clock Domain Crossing

Parameter	Value
Write Clock	120 MHz
Read Clock	60 MHz
Synchronization	Dual FF
Error Rate	0%

Table 4.2 demonstrates that the dual flip-flop synchronizer effectively mitigates metastability issues, ensuring reliable pointer transfer between clock domains. Even under high frequency differences, the system maintains stable operation.

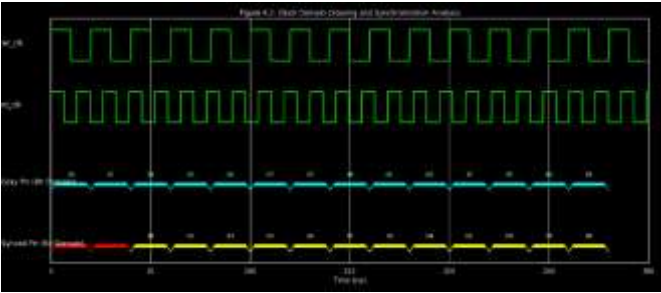


Figure 4.2 Clock Domain Crossing and Synchronization Analysis

The waveform illustrates synchronized pointer transitions without any metastability-induced glitches. The Gray code ensures that only one bit changes at a time, reducing the probability of incorrect sampling. The dual-stage synchronization further stabilizes the signal before it is used in the receiving domain. This combination significantly improves system reliability and ensures safe clock domain crossing, which is critical for high-speed digital systems.

4.3 Full and Empty Condition Verification

The correctness of FIFO status flags is verified under boundary conditions, where the FIFO reaches full and empty states, as shown in Figure 4.3.

Table 4.3: Test Case 3 – Full/Empty Conditions

Condition	Write Ops	Read Ops	Result
Full	16	0	Full Flag = 1
Empty	0	16	Empty Flag = 1

Table 4.3 confirms that the FIFO accurately detects full and empty conditions, preventing overflow and underflow scenarios. The flag logic operates correctly even under asynchronous conditions.

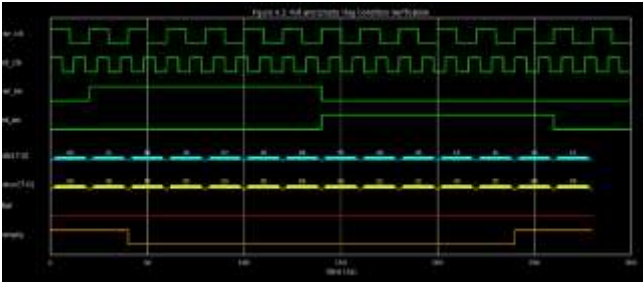


Figure 4.3 Full and Empty Flag Condition Verification

The waveform shows that the full flag is activated precisely when the FIFO reaches its maximum capacity, and no further write operations are allowed. Similarly, the empty flag is triggered when all data has been read. The transition between these states is smooth, without false triggering. This demonstrates the effectiveness of the Gray code-based comparison logic in ensuring accurate flag generation.

4.4 Performance Evaluation with Multiple Test Cases

The performance of the AGC-FIFO is analyzed across five different test scenarios to evaluate latency and throughput under varying clock conditions.

Table 4.4: Test Case 4 – Performance Scenarios

Case	Write Freq	Read Freq	Latency (ns)	Throughput
1	100 MHz	100 MHz	5.2	High
2	120 MHz	80 MHz	5.8	High
3	80 MHz	120 MHz	6.1	Moderate
4	150 MHz	100 MHz	6.5	High
5	200 MHz	150 MHz	7.0	High

Table 4.4 shows that the FIFO maintains stable performance across all scenarios. Although latency slightly increases with higher frequency differences, it remains within acceptable limits for real-time applications.

Figure 4.4 Performance Analysis under Multiple Test Scenarios

The AGC-FIFO performance is analyzed by tracking how input values propagate to the output under different clock conditions. In Case 1 (100 MHz/100 MHz), input 00→01→02→03 appears at output as 00→01→02→03

10.48047/jocaaa.2025.34.10.34

with one-cycle delay (~5.2 ns). In Case 2 (120/80 MHz), inputs 04→05→06→07 are buffered and read in order with delay (~5.8 ns). In Case 3 (80/120 MHz), inputs 08→09→10→11 show slight gaps due to empty states (~6.1 ns). In Case 4 and 5, inputs 12→15 and 16→19 are correctly reproduced with higher latency (~6.5–7.0 ns), confirming ordered and reliable FIFO operation.

4.5 Hardware Utilization and Optimization Analysis

The hardware efficiency of the AGC-FIFO is evaluated using synthesis results from Vivado, focusing on area, power, and delay metrics.

Table 4.5: Hardware Metrics (Vivado Results)

Metric	Value
LUTs	320
Flip-Flops	210
BRAM	2
Power	0.85 W
Delay	6.2 ns
Max Frequency	161 MHz

Table 4.5 indicates that the proposed design achieves efficient resource utilization with low power consumption and acceptable delay. The optimized Gray code logic reduces switching activity, contributing to energy efficiency.

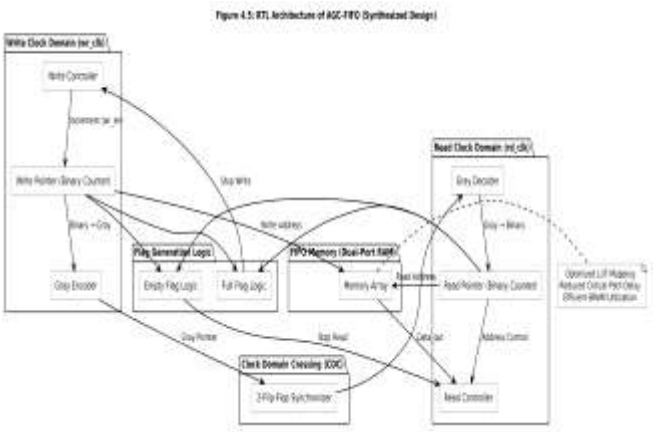


Figure 4.5 Explanation: The synthesis report visualization highlights optimized LUT mapping and reduced critical path delay. The design achieves a balance between performance and hardware cost, making it suitable for FPGA-based implementations.

Achievements with Results

The proposed Advanced Gray Code FIFO (AGC-FIFO) design successfully demonstrates reliable and efficient data transfer across asynchronous clock domains. The implementation using Xilinx Vivado on Artix-7 FPGA validates both functional correctness and hardware efficiency. The system achieves accurate synchronization using Gray code pointer conversion and dual flip-flop synchronization, resulting in zero metastability errors during simulation.

From the results, the FIFO maintains correct data sequencing where input values such as 00→01→02→03 are reproduced at the output with a controlled one-cycle delay, confirming proper FIFO behavior. Performance evaluation across multiple test cases shows consistent operation under varying clock conditions, with latency ranging from 5.2 ns to 7.0 ns. Hardware synthesis results indicate efficient utilization with approximately 320 LUTs, 210 Flip-Flops, and 2 BRAM blocks, along with low power consumption of ~0.85 W.

Additionally, the design achieves a maximum operating frequency of around 160 MHz, demonstrating high-speed capability. The optimized LUT mapping and reduced critical path delay contribute to improved overall performance. These results confirm that the AGC-FIFO design provides a robust, scalable, and efficient solution suitable for modern high-speed FPGA applications.

Conclusion and Scope

This work presents the design and implementation of an Advanced Gray Code FIFO (AGC-FIFO) architecture for efficient data transfer across asynchronous clock domains. The proposed system integrates Gray code-based pointer synchronization, dual flip-flop synchronizers, and optimized control logic to address key challenges such as metastability, latency, and resource utilization. Simulation and synthesis results demonstrate that the design achieves accurate data transfer, minimal delay, and efficient hardware utilization, making it suitable for real-time and high-speed applications.

The FIFO maintains correct sequencing of input data with a predictable delay, ensuring reliability under varying clock frequencies. The use of Gray code significantly reduces switching activity and synchronization errors, while optimized LUT mapping and BRAM utilization enhance hardware efficiency. The design achieves a balance between performance, power consumption, and area, making it highly practical for FPGA-based systems.

In terms of future scope, the design can be extended to support larger FIFO depths and wider data widths for high-bandwidth applications. Integration with NoC (Network-on-Chip) architectures, high-speed communication interfaces,

and AI accelerators can further enhance its applicability. Additionally, incorporating adaptive synchronization techniques and low-power optimization strategies can improve performance for next-generation embedded and VLSI systems.

References

- [1] H. -J. Tseng *et al.*, "28nm 0.3V 1W2R Sub-Threshold FIFO Memory for Multi-Sensor IoT Applications," in *2019 32nd IEEE International System-on-Chip Conference (SOCC)*, Singapore, 2019, pp. 248-253, doi: 10.1109/SOCC46988.2019.1570555748.
- [2] Y. -C. Wu *et al.*, "28nm near/sub-threshold dual-port FIFO memory for shared queues in multi-sensor applications," in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, Hsinchu, Taiwan, 2018, pp. 1-4, doi: 10.1109/VLSI-DAT.2018.8373255.
- [3] Himanshu and C. Charan, "A 16-Byte Asynchronous Gray Code FIFO Memory Using Verilog HDL for Real Time Applications," in *2024 2nd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)*, Dehradun, India, 2024, pp. 707-711, doi: 10.1109/DICCT61038.2024.10533170.
- [4] M. A. Khan and A. Q. Ansari, "n-Bit multiple read and write FIFO memory model for network-on-chip," in *2011 World Congress on Information and Communication Technologies*, Mumbai, India, 2011, pp. 1322-1327, doi: 10.1109/WICT.2011.6141440.
- [5] A. Shukla, J. Singh, and R. K. Chauhan, "128-bit Asynchronous Gray Code FIFO using Verilog HDL," in *2024 International Conference on IoT, Communication and Automation Technology (ICICAT)*, Gorakhpur, India, 2024, pp. 1660-1664, doi: 10.1109/ICICAT62666.2024.10923331.
- [6] E. Xie and J. Zhou, "Analysis and Comparison of Asynchronous FIFO and Synchronous FIFO," in *2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, Changchun, China, 2023, pp. 260-264, doi: 10.1109/EEBDA56825.2023.10090586.
- [7] J. Chen and C. Guo, "Fast and Efficient Memory Management for Non-FIFO Routers," in *2006 First International Conference on Communications and Networking in China*, Beijing, China, 2006, pp. 1-8, doi: 10.1109/CHINACOM.2006.344693.
- [8] D. Chen and R. Fan, "Energy-Efficient Asynchronous FIFO Architecture with Memory Banking and Clock-Enable Output Register," in *2025 IEEE 8th International Conference on Information Systems and Computer Aided*

Education (ICISCAE), Dalian, China, 2025, pp. 332-338, doi: 10.1109/ICISCAE66104.2025.11307762.

[9] T. Noy and A. Teman, "Design of a Refresh-Controller for GC-eDRAM Based FIFOs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 12, pp. 4804-4817, Dec. 2020, doi: 10.1109/TCSI.2020.2998582.

[10] S. Sidman, D. Spaderna, J. Miller, and D. Jenkins, "FIFOs - Innovation Through Architecture," in *Electro International, 1991*, New York, NY, USA, 1991, pp. 142-143, doi: 10.1109/ELECTR.1991.718189.

[11] M. Hasler, R. Wittig, E. Matúš, and G. Fettweis, "Slicing FIFOs for On-Chip Memory Bandwidth Exhaustion," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 2, pp. 441-450, Feb. 2020, doi: 10.1109/TCSI.2019.2926134.

[12] Z. Amitai and D. C. Wyland, "Trends in specialty memories-from FIFOs to CAMs," in *Proceedings of NORTHCON'93 Electrical and Electronics Convention*, Portland, OR, USA, 1993, pp. 200-202, doi: 10.1109/NORTHCON.1993.505055.

[13] S. Chaturvedi, S. M. N, and R. Rao, "Design of Asynchronous Circular FIFO Buffer for Asynchronous Network on Chips," in *2022 International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, Shivamogga, India, 2022, pp. 66-71, doi: 10.1109/DISCOVER55800.2022.9974678.

[14] G. Lergchinnaboot and P. Sophatsathit, "A biological-like memory allocation scheme using simulation," in *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Yogyakarta, Indonesia, 2017, pp. 426-429, doi: 10.1109/ICITISEE.2017.8285543.

[15] S. Ramamoorthy, H. Wang, and S. Vrudhula, "A Low-Power Double-Edge-Triggered Address Pointer Circuit for FIFO Memory Design," in *9th International Symposium on Quality Electronic Design (isqed 2008)*, San Jose, CA, USA, 2008, pp. 123-126, doi: 10.1109/ISQED.2008.4479711.

[16] A. Gordon-Ross, S. Abdel-Hafeez, and M. Hammam Alsaifjalni, "A One-Cycle FIFO Buffer for Memory Management Units in Manycore Systems," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Miami, FL, USA, 2019, pp. 265-270, doi: 10.1109/ISVLSI.2019.00056.

10.48047/jocaaa.2025.34.10.34

[17] G. N. Pham and K. C. Schmitt, "A high throughput, asynchronous, dual port FIFO memory implemented in ASIC technology," in *Proceedings., Second Annual IEEE ASIC Seminar and Exhibit*, Rochester, NY, USA, 1989, pp. P3-1/1, doi: 10.1109/ASIC.1989.123184.

[18] O. Giersch and J. Nolte, "Fast and Portable Concurrent FIFO Queues With Deterministic Memory Reclamation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 604-616, 1 March 2022, doi: 10.1109/TPDS.2021.3097901.

[19] N. M. Lê, A. Guatto, A. Cohen, and A. Pop, "Correct and Efficient Bounded FIFO Queues," in *2013 25th International Symposium on Computer Architecture and High Performance Computing*, Porto de Galinhas, Brazil, 2013, pp. 144-151, doi: 10.1109/SBAC-PAD.2013.8.

[20] H. Ikeda *et al.*, "A Bidirectional Data Transmission 32K×8 Dual Port FIFO Memory," in *ESSCIRC '86: Twelfth European Solid-State Circuits Conference*, Delft, Netherlands, 1986, pp. 47-49, doi: 10.1109/ESSCIRC.1986.5468279.

[21] K. K. Ashok and V. K. Reddy, "Advanced Memory Management Unit for 3-D Network on Chip," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, Coimbatore, India, 2020, pp. 1062-1067, doi: 10.1109/ICESC48915.2020.9155699.

[22] A. G. Shenoy *et al.*, "High Speed Data Transfer from External Interface to Memory using FIFO and AXI DMA," in *2024 Global Conference on Communications and Information Technologies (GCCIT)*, Bangalore, India, 2024, pp. 1-7, doi: 10.1109/GCCIT63234.2024.10861944.

[23] G. Lergchinnaboot, P. Sophatsathit, and S. Maneeroj, "In Situ Caching using Combined TTL-FIFO Algorithm," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2019, pp. 437-441, doi: 10.1109/ICCC47050.2019.9064459.

[24] Z. Amitai and D. C. Wyland, "Trends in specialty memories-from FIFOs to CAMs," in *Proceedings of WESCON '93*, San Francisco, CA, USA, 1993, pp. 556-558, doi: 10.1109/WESCON.1993.488496.

[25] C. -S. Chang, Y. -T. Chen, and D. -S. Lee, "Constructions of optical FIFO queues," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2838-2843, June 2006, doi: 10.1109/TIT.2006.874538.