Vol. 16 (2026): Proceedings of the 2026 International Conference on AI and Technological Innovation Applications (AITIA 2026)
E-ISSN: 3105-0522 P-ISSN: 3105-0514
DOI: https://doi.org/10.70267/aitia.20267480

Design of Pre-reading Asynchronous FIFO Based on Verilog

Chenhao Huang*

School of Electronic Science and Engineering(School of Microelectronics), South China Normal University, Foshan 528225, China

**Corresponding author: Chenhao Huang.*

Abstract

With the rapid advancement of modern Very Large Scale Integration (VLSI), the increasing complexity of System-on-Chip (SoC) designs has made traditional single-clock domain synchronous architectures insufficient for meeting the escalating demands of high-performance computing and high-bandwidth communication. As a result, asynchronous FIFO has become increasingly prevalent for facilitating Clock Domain Crossing (CDC) data transfer, ensuring reliable communication between asynchronous interfaces. However, conventional asynchronous FIFOs suffer from inherent latency during the data transmission process, leading to a significant lag in data read-out, which ultimately compromises overall system efficiency. Based on Verilog HDL, this paper explores the optimization of standard asynchronous FIFO architectures. By reconfiguring the internal control logic, the design successfully incorporates First-Word Fall-Through (FWFT) functionality, effectively eliminating traditional read-to-output latency and minimizing data retrieval delays, thereby significantly enhancing transmission efficiency and throughput.

Keywords

asynchronous FIFO, first-word fall-through, FIFO design

1. Introduction

With the rapid advancement of integrated circuit (IC) fabrication technology, the scale and architectural complexity of modern Systems-on-Chip (SoC) designs have increased significantly [1], which consequently exacerbates the performance and power constraints of foundational circuit elements in advanced technology nodes [2]. Under such stringent physical constraints, distributing a single high-frequency global clock across the entire chip becomes highly impractical due to severe clock skew and excessive dynamic power consumption. To strike a balance between high performance and low power consumption, complex integrated circuits often adopt the Globally Asynchronous Locally Synchronous (GALS) architecture. Under this paradigm, functional blocks such as CPU cores, caches, memory controllers, and various peripheral bus modules typically operate at different clock frequencies [3]. As a critical component for data transmission across distinct clock domains (CDC), the asynchronous FIFO has demonstrated increasing utility and a broadening scope of adoption in modern digital system design [4]. For instance, the Tesla D1 processor adopts the GALS architecture to mitigate clock skew challenges inherent in its large-scale design (Total Die Area 645

mm²) [5]. Intel Labs' Loihi chip demonstrates that asynchronous communication interfaces are critical for processing the bursty and non-synchronous neural spikes inherent in neuromorphic computing [6]. AMD extensively utilizes synchronizing FIFO within the Infinity Fabric architecture of its Zen series processors to facilitate data exchange across disparate chiplets [7]. While conventional asynchronous FIFOs are widely used for clock domain crossing in data transmission, their design logic imposes a significant limitation in practical engineering applications: reading latency. In standard asynchronous FIFOs, data typically appears on the output bus only at the subsequent rising edge after the Read Enable signal is asserted. This characteristic leads to a series of issues, including reduced pipeline efficiency, loss of real-time performance, and increased complexity in protocol interfacing. Research into First-Word Fall-Through (FWFT) technology is specifically aimed at eliminating this single-clock-cycle delay, allowing the first valid data to automatically "fall through" to the output end, thereby achieving a logically "zero-latency" read. This is of great engineering significance for optimizing bus handshake efficiency and simplifying the design of downstream modules. The subject of this paper is the design and verification of an FWFT asynchronous FIFO. The research focus lies in how to achieve FWFT characteristics through logic reconfiguration while ensuring the robustness of asynchronous signal synchronization and avoiding metastability in multi-clock domain environments. The core of the research lies in the conversion and synchronization of asynchronous pointers and the determination of Empty/Full flags.

In this paper, theoretical analysis, RTL modeling, and simulation verification are combined to design and evaluate an asynchronous FWFT FIFO. The proposed design is implemented in Verilog HDL and adopts a modular architecture that separates the pointer control logic, storage array, and FWFT control logic. This structure improves design clarity and facilitates functional verification. The main objective is to realize a high-performance and reliable asynchronous FIFO in which the first valid data is available on the output bus before the read enable signal is asserted. As a result, the proposed FWFT mechanism reduces read latency and enhances data transmission efficiency across different clock domains.

2. Theoretical Basis of Pre-reading Asynchronous FIFO

2.1 Fundamental Principles of Asynchronous FIFO

The asynchronous FIFO consists of six modules in total. The data read module and the data write module determine whether to perform read or write operations based on the clock of the corresponding read/write region. Upon the rising edge of the write clock, data is written to the dual-port random access memory at the write address, provided that the write enable signal is asserted and the FIFO is not full. Data reading is performed at the rising edge of the read clock, where the stored data are retrieved from the address indicated by the read pointer, provided that the read enable signal is valid and the FIFO is not empty. The address Gray code conversion module is employed to convert the binary addresses of the read and write pointers into Gray code, thereby facilitating the comparison of their positional relationship. The Gray code synchronization module mitigates the challenges associated with clock domain crossing. The Gray-coded read pointer is synchronized into the write clock domain. In contrast, the Gray-coded write pointer is synchronized into the read clock domain, thereby enabling the comparison of the read and write addresses. The empty/full detection module determines the status of the asynchronous FIFO by comparing the synchronized Gray codes, thereby governing the permission of read and write operations. The data storage module (dual-port random access memory) is responsible for storing data written into the FIFO, while simultaneously enabling data retrieval from the FIFO.

2.2 Key Technologies in Asynchronous FIFO Design

The design of an asynchronous FIFO is mainly characterized by two critical challenges: the accurate determination of empty and full conditions from the read and write pointer states, and the realization of reliable data transmission across different clock domains.

The determination of the FIFO's empty and full conditions is contingent upon comparing the positional relationship between the read and write pointers. However, the read and write pointers coincide in both empty and full states, making it difficult to distinguish between these two conditions. To resolve this issue, the bit-width of the read and write pointers is expanded by adding a bit to each pointer [8]. This allows for an

unambiguous determination of the FIFO's empty or full status based on the relative positions of the read and write pointers. As the read and write pointers of an asynchronous FIFO reside in distinct clock domains, their addresses cannot be directly compared. Consequently, Cross-Clock Domain data transfer is required, necessitating robust mechanisms to prevent metastability during transmission [9].

Table 1: FIFO Pointer Comparison Logic for Status Generation

Read the pointer in Gray code	Write the pointer in Gray code	FIFO Status
0000	0000	empty
0000	1100	full
1100	0000	full

To facilitate cross-clock domain data transfer, a two-stage synchronizer is required in each of the read and write clock domains. Table 1 illustrates the pointer comparison logic used for FIFO status generation. In an asynchronous FIFO, the read and write pointers belong to different clock domains and therefore cannot be directly compared before synchronization. To ensure reliable status judgment, the Gray-coded write pointer is synchronized into the read clock domain for empty detection, while the Gray-coded read pointer is synchronized into the write clock domain for full detection [10]. The empty condition occurs when the synchronized write pointer and the current read pointer are identical, indicating that no unread data remains in the FIFO. In contrast, the full condition occurs when the two most significant bits of the Gray-coded pointers are inverted while the remaining lower bits remain the same, which indicates that the write pointer has caught up with the read pointer after one complete address cycle. By using Gray-coded pointers, only one-bit changes between adjacent pointer values, thereby reducing the risk of metastability during cross-clock-domain synchronization and improving the reliability of empty/full flag generation.

2.3 Delay Analysis of Asynchronous FIFO

There are numerous factors that contribute to the latency of an asynchronous FIFO. The information transfer of the read and write pointers across clock domains requires synchronizers, which introduces a latency of several clock cycles. The empty/full determination in an asynchronous FIFO follows a conservative logic; even when the FIFO is not actually empty, the empty flag may remain asserted for a while. Similarly, even when the FIFO is not actually full, the full flag may remain asserted for a while. The reset and startup phases can also introduce brief delays. However, these delays are all necessary to prevent metastability in the asynchronous FIFO and to avoid misjudging the empty/full status; they are not design defects. But in a conventional asynchronous FIFO, data retrieval from the dual-port random access memory is contingent upon the read enable signal being active, which incurs a latency of at least one clock cycle. This part of the delay can be mitigated through design.

2.4 Principle of Pre-reading Mechanism in Asynchronous FIFO

In conventional asynchronous FIFO architectures, data retrieval is strictly contingent upon the simultaneous assertion of the read enable signal and a non-empty status. This dependency on the read-enable handshake introduces an inherent latency that can bottleneck high-speed systems. To mitigate this delay, a structural redesign is required to decouple data availability from explicit control assertions. Achieving pre-reading capabilities requires disregarding the read enable signal during data retrieval. With the FIFO empty flag serving as the exclusive control condition, the read enable signal is utilized solely to advance the read pointer. If the read clock rises and the FIFO is not empty, data can be retrieved immediately. This mechanism effectively eliminates the latency associated with waiting for the read enable signal.

3. Design and Simulation Verification

3.1 Verilog Hardware Modeling

Figure 1 illustrates the overall architecture of the First-Word Fall-Through (FWFT) asynchronous FIFO. The design comprises a write controller, a read controller, a dual-port RAM, write-to-read (W2R) and read-to-write (R2W) synchronizers, and an output pipeline stage. The write controller operates within the write clock domain to generate the write address and write pointer, whereas the read controller functions within the read clock domain to generate the read address and read pointer. To ensure reliable cross-clock domain

communication, the read/write control logic relies on the W2R and R2W synchronizers to precisely synchronize the pointers across the two clock domains: the write pointer is synchronized to the read clock domain via the W2R synchronizer to perform empty detection, while the read pointer is synchronized to the write clock domain via the R2W synchronizer for full detection, thereby guaranteeing the reliability of cross-clock-domain data accesses.The dual-port RAM stores the incoming data and supports independent read and write operations across distinct clock domains, thereby enabling parallel access.

Figure 1: Block diagram of the FTFW asynchronous FIFO architecture

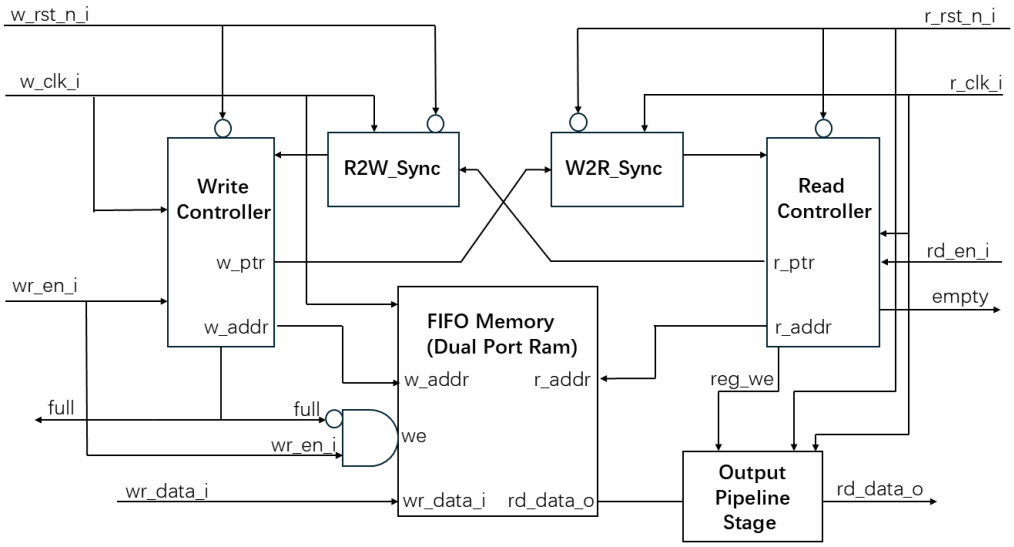


Table 2: Signal Names and Descriptions

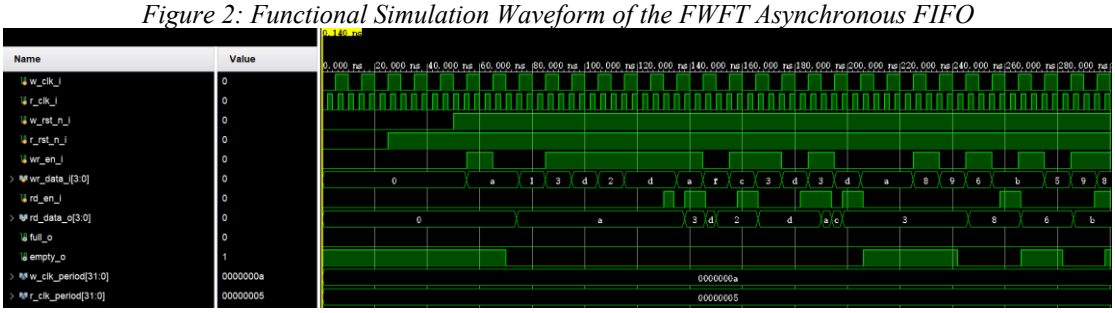
Signal name	Functional Description
w_clk_i	Input write clock
r_clk_i	Input read clock
w_rst_n_i	Input write reset (active low)
r_rst_n_i	Input read reset (active low)
wr_en_i	Input write enable
rd_en_i	Input read enable
wr_data_i	Input write data
rd_data_o	Output read data
reg_we	Write enable signal for output register

Table 2 summarizes the main interface signals of the proposed FWFT asynchronous FIFO architecture, including the write clock domain, read clock domain, data interface, control signals, and internal register control signal. These signals jointly define the basic input and output behavior of the FIFO and provide the necessary interfaces for data storage, data retrieval, and output register control. The signals w_clk_i and r_clk_i are the independent write and read clocks, respectively, enabling the FIFO to operate across two different clock domains, while the active-low reset signals w_rst_n_i and r_rst_n_i are used to initialize the write and read domains and bring the internal pointers, status flags, and related control logic into a known state after reset. For data transmission, the interface utilizes wr_data_i and rd_data_o as the input and output data buses, where wr_data_i is used to write data into the FIFO and rd_data_o is used to output the stored data to the downstream logic. The operation is managed by the write enable signal wr_en_i and the read enable signal rd_en_i, where wr_en_i determines whether a write operation is permitted in the write clock domain and rd_en_i controls the update of the read pointer in the read clock domain. In addition, the reg_we signal serves as the write enable control for the output register and is used to control the timing of data output in the FWFT mechanism, thereby ensuring that valid data can be presented on the output bus in a timely manner.

3.2 Functional Simulation

As illustrated in Figure 2, the write clock w_clk_i operates at 100 MHz, while the read clock r_clk_i is configured at 200 MHz, forming an asynchronous clock environment with a 2:1 frequency ratio between the two domains. This setup is a typical scenario for evaluating the robustness of an asynchronous FIFO. The reset signals are de-asserted between 30 ns and 40 ns, after which both clock domains enter normal operation. The stability of the clock waveforms after reset indicates that the clock generation and reset release mechanisms are functioning correctly. At approximately 60 ns, the write enable signal is asserted, and data begins to be written sequentially into the FIFO on the rising edge of the write clock. The waveform shows that the write data changes in a controlled manner, confirming successful write operations into the buffer. Shortly after, at around 70 ns, although the read enable signal remains de-asserted, the read data output has already transitioned to 'a'. This verifies that the FIFO correctly implements the First-Word Fall-Through (FWFT) feature, meaning that the first stored word becomes available at the output immediately after being written, without requiring an explicit read request. In other words, the read path does not need to wait for the read enable signal to present the first data word, which reduces access latency and improves throughput. After the write operation, the empty flag transitions from 1 to 0, indicating that the FIFO is no longer empty once valid data has been stored. Meanwhile, the full flag remains at 0 throughout the simulation, showing that the FIFO never reaches its maximum storage capacity under the applied test sequence. This confirms that the full and empty flag generation logic is operating as expected. As additional read and write operations continue, the read data output updates accordingly, demonstrating that data is retrieved in the correct order and that the FIFO preserves the expected first-in, first-out behavior. When the next rising clock edge arrives, the read output transitions from 'a' to the subsequent value '3', indicating that the read pointer advances only when necessary. This confirms that, in an FWFT asynchronous FIFO, the read enable signal is not used to request the first available data word, since that data is already present at the output; instead, it is used to advance the read pointer to the next stored location. Overall, the simulation results verify that the FIFO operates correctly across asynchronous clock domains, with proper synchronization, correct flag behavior, and successful FWFT functionality.

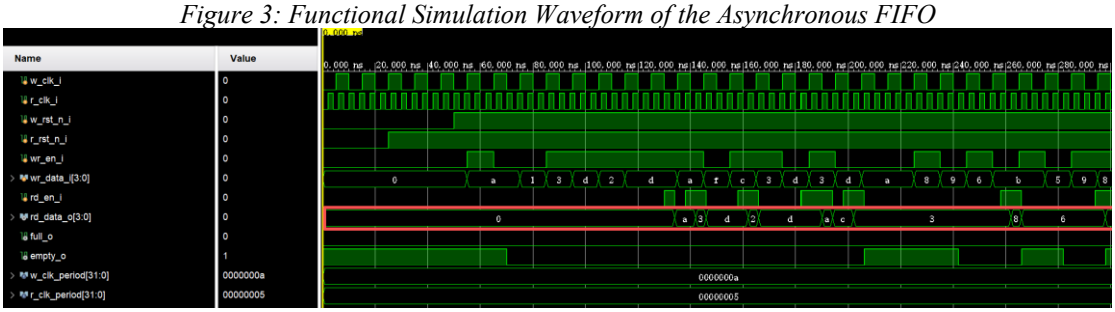
Figure 2: Functional Simulation Waveform of the FWFT Asynchronous FIFO



3.3 Comparison with Conventional Asynchronous FIFO

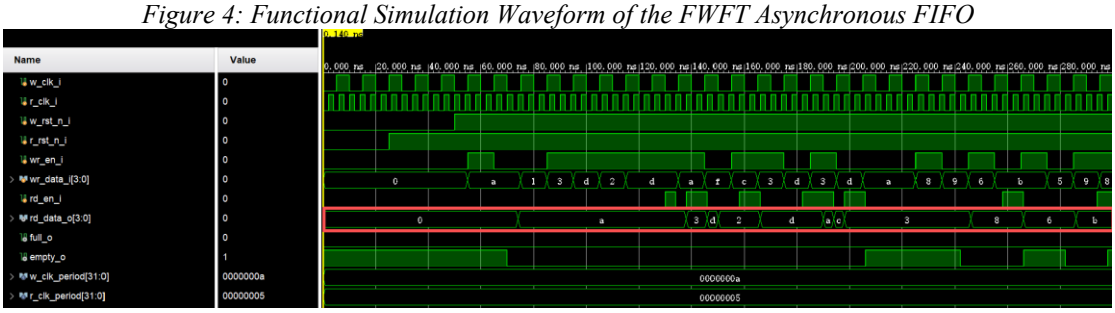
As shown in Figures 3 and 4, compared with conventional non-FWFT (standard) FIFO architectures, the proposed design demonstrates significant performance advantages in terms of data throughput and latency reduction templates. First, in terms of read latency, the proposed FIFO effectively eliminates the traditional one-cycle read-to-output delay. As illustrated in Figure 3, at approximately 134 ns, the conventional asynchronous FIFO exhibits a distinct latency, as data 'a' only reaches the output bus rd_data_o after a one-clock-cycle delay following the assertion of the read enable signal.

Figure 3: Functional Simulation Waveform of the Asynchronous FIFO



In contrast, the waveform in Figure 4 at approximately 70 ns confirms that data 'a' arrives at the output bus rd_data_o immediately after being written, without waiting for the read enable signal. Further comparison between Figures 3 and 4 reveals that at the 140 ns and 160 ns marks, the data read operations in Figure 4 are significantly faster than those in Figure 3. This demonstrates that, compared with the conventional non-FWFT (standard) FIFO, the proposed FIFO indeed reduces latency. This zero-cycle latency proves that the FWFT mechanism successfully bypasses the conventional read-handshake overhead, thereby drastically minimizing idle clock cycles. Second, regarding the reliability of status flags, the de-assertion of the empty_o signal is perfectly synchronized with the availability of data at the output port. This indicates that the internal empty/full logic is robust even under the demanding 1:2 write-to-read clock ratio (100 MHz write vs. 200 MHz read). Unlike standard FIFOs, where the read enable signal acts as a trigger to pull data, here it functions strictly as a pointer incrementer, allowing for continuous and streamlined data stream processing that improves overall timing alignment between disparate clock domains. Consequently, the design not only ensures functional correctness but also achieves superior data alignment and synchronization efficiency, confirming its suitability for high-performance, low-latency asynchronous data processing tasks.

Figure 4: Functional Simulation Waveform of the FWFT Asynchronous FIFO



4. Conclusion

In this paper, a high-performance asynchronous FWFT FIFO has been successfully designed and verified. Experimental and simulation results demonstrate that the proposed architecture achieves significant performance advantages over conventional standard FIFOs in terms of data throughput and latency reduction. By fully leveraging the FWFT mechanism, the design successfully eliminates the traditional one-cycle read-to-output delay, realizing “zero-cycle” data availability and minimizing invalid wait cycles. Furthermore, the status logic (empty/full flags) exhibits excellent timing robustness and functional correctness, even under highly stressed and asymmetric clock ratios (e.g., 1:2). With modern SoCs and multi-core processors migrating toward higher operating frequencies and complex heterogeneous integration, CDC has become a critical bottleneck for low-latency communication. The proposed asynchronous FWFT FIFO offers an efficient hardware solution for high-speed, low-latency data alignment. It holds significant potential for integration into real-world high-throughput applications, such as Network-on-Chips (NoCs), high-speed serial interfaces (e.g., PCIe), and memory controllers. The design contributes valuable insights to enhancing system-level timing margins and overall data transmission efficiency. Although the proposed design achieves the anticipated performance objectives, there remains scope for further optimization. Future research could focus on incorporating hardware-level low-power design techniques, such as clock gating, to further suppress dynamic power consumption under high-frequency operation. Additionally, subsequent work will involve ASIC synthesis and backend physical layout design to evaluate the practical silicon area and power dissipation.

References

[1] Chen, Z., et al. (2023). AI SoC design challenges in the foundation model era. In 2023 IEEE Custom Integrated Circuits Conference (CICC) (pp. 1–8). IEEE. https://doi.org/10.1109/CICC57935.2023.10121242

[2] Verbitsky, D., Dobkin, R. R., Ginosar, R., et al. (2014). StarSync: An extendable standard-cell mesochronous synchronizer. Integration, the VLSI Journal, 47(2), 250–260. https://doi.org/10.1016/j.vlsi.2013.09.003

[3] Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014). The SpiNNaker project. Proceedings of the IEEE, 102(5), 652–665. https://doi.org/10.1109/JPROC.2014.2304638

[4] Wang, Z. (2024). Research progress of asynchronous FIFO design. Science and Technology of Engineering, Chemistry and Environmental Protection, 1(6). https://doi.org/10.61173/kjgkj796

[5] Talpes, E., et al. (2023). The microarchitecture of DOJO, Tesla's exa-scale computer. IEEE Micro, 43(3), 31–39. https://doi.org/10.1109/MM.2023.3258906

[6] Davies, M., et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro, 38(1), 82–99. https://doi.org/10.1109/MM.2018.112130359

[7] Singh, T., et al. (2020). 2.1 Zen 2: The AMD 7nm energy-efficient high-performance x86-64 microprocessor core. In 2020 IEEE International Solid-State Circuits Conference - (ISSCC) (pp. 42–44). IEEE. https://doi.org/10.1109/ISSCC19947.2020.9063113

[8] Tammisetti, D., Jyothika, K., Rajesh, P. N. V., Devi, M. N., Reddy, K. M., & Raju, P. P. (2025). Design and verification of a parameterized asynchronous FIFO using Gray code synchronization. IJIRT, 12(9), 2926–2935.

[9] Zhang, X. (2024). Optimizing data transfer across asynchronous clock domains: A comprehensive approach to asynchronous FIFO circuit design. Highlights in Science, Engineering and Technology, 87, 31–36. https://doi.org/10.54097/ab66rm52

[10] Konstantinou, D., Psarras, A., Nicopoulos, C., & Dimitrakopoulos, G. (2020). The mesochronous dual-clock FIFO buffer. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28(1), 302–306. https://doi.org/10.1109/TVLSI.2019.2946348

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.

Open Access