VIDWAN Journal of Scientific Research

OPEN ACCESS

PEER-REVIEWED - OPEN ACCESS Vol. 01 · Issue 01 · 2026 · pp. 13-20 | ISSN: APPLIED (Online) | INAGURAL EDITION

Design And Performance Analysis Of An Asynchronous FIFO Using Verilog

R. Aswitha, *Department of CSE,*
Gurunanak Institute of Technology(Gnit), Hyderabad.

ABSTRACT

The First-In, First-Out (FIFO) method is a popular technique for managing computer work requests that originate from stacks or queues. Collectively, these measures ensure that the initial request receives the attention it warrants. Data is transmitted between clock domains with FIFO (first-in, first-out) logic. This operation is executed in hardware by a sequence of read/write memory components or flip-flops. Enhancing the development of a First-In-First-Out (FIFO) system can be achieved by comparing write and read pointers generated in distinct clock domains rather than simultaneously. Asynchronous First-In-First-Out (FIFO) systems employ pointer comparison to minimize the quantity of synchronization flip-flops required for FIFO construction. This academic paper elucidates the necessity of supplementary methods for the efficient construction and assessment of designs inside this framework. This design enhances the efficiency of the First-In-First-Out (FIFO) approach by employing mixed binary/gray counters that utilize integrated binary ripple carry circuitry.

Keywords: Asynchronous FIFO, FIFO Design, Full and Empty Deduction.

Article history: Received 8 July 2026 · Revised 18 July 2026 · Accepted 28 July 2026 · Published 7 August 2026

1.INTRODUCTION

A solitary clock domain is employed to progressively input data values into a First-In-First-Out (FIFO) buffer. Employing an independent clock domain facilitates the sequential reading of data values from a common FIFO buffer. Neither of these devices possesses a synchronized clock domain. Gray code pointers are occasionally employed in the construction of an asynchronous First-In-First-Out (FIFO) system. Prior to indicating the FIFO's full or empty status using synchronous signals, these pointers are synchronized to the appropriate time domain. A novel and intriguing approach to ascertain FIFO fullness and vacancy indicators is to concurrently compare pointers prior to designating full or empty status bits.

Full and Empty Deductions

Ensuring the right functionality of both full and empty states is the most challenging aspect of creating a First-In-First-Out (FIFO) system. An additional ingredient is required to distinguish between a state of fullness and a state of emptiness. The method divides the address space into four quadrants, utilizing the most significant bits (MSBs) of the two counters to decode data when two points possess identical values. This demonstrates the extent to which the First-In-First-Out (FIFO) system is either full or empty.

The write pointer (wptr) is positioned one quadrant ahead of the read pointer (rptr), making the First-In-First-Out (FIFO) method the most efficient approach. When the write pointer is one region behind the read pointer, the memory approaches its maximum capacity. Select the directional icon to initiate the designated action.

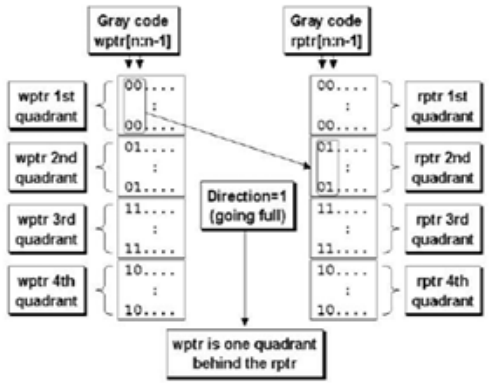


Fig-1: The capacity of the first-in, first-out (FIFO) method is limited because the read pointer (rptr) is one quadrant behind the write pointer (wptr).

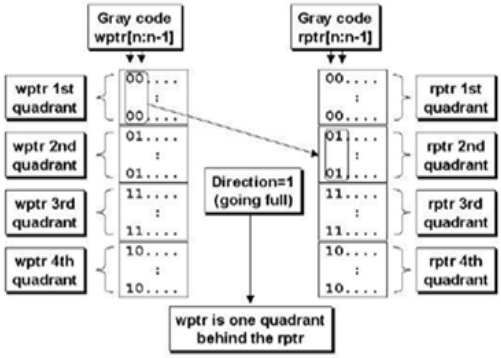


Fig-2: Because the write pointer (wptr) is placed one quadrant behind the read pointer (rptr), the First-In-First-Out (FIFO) mechanism works efficiently.

When the system indicates that the situation may be vacant and the write pointer is one quadrant ahead of the read pointer, the direction latch remains unlocked.

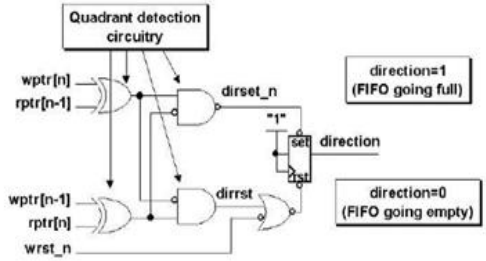


Fig -3: Because the write pointer (wptr) is one quadrant after the read pointer (rptr), the First-In-First-Out (FIFO) mechanism is fully operational.

To ascertain whether the first-in, first-out (FIFO) structure is vacant post-reset, one may clear the direction latch. The address identification decoder's directional switching capability eliminates any ambiguity. Consequently, configuring and deactivating the directional closure is superfluous. Two 4-input lookup tables suffice to construct the Xilinx FPGA circuitry that decodes the two most significant bits of the wptr and rptr. The intrinsic inconsistency of write and read times gives rise to a second, more significant issue. Decoding leaps may be erroneous when numerous bits in both counts change sequentially. The paper's findings indicate that the disparity between consecutively assessed gray count sequences is merely one bit. Decoders and comparators exclusively transition between optimal outputs, resulting in error-free decoding.

FIFO2.v

This module comprehensively addresses all clock domains. All supplementary FIFO modules required for the construction plan are constructed upon the primary module. In a larger Application-Specific Integrated Circuit (ASIC) or Field-Programmable Gate Array (FPGA) design, the comprehensive wrapper would likely be eliminated if this First-In-First-Out (FIFO) structure were employed. This technique will enhance the synthesis

and static timing analysis processes by optimizing the classification of the remaining First-In-First-Out (FIFO) modules into their appropriate clock domains.

FIFO mem.v

The FIFO memory buffer is utilized by the read and write clock domains. This buffer can be referred to as dual-port synchronous RAM. A variety of memory types can interface with the FIFO buffer.

async_cmp.v

The signals from the asynchronous pointer-comparison module denote the regulated full and vacant state bits. These signals are programmed to activate at predetermined periods. This section is characterized by a significant dependence on combinatorial comparison logic. None of the included programs employs sequential logic.

rptr_empty.v

This module encompasses the code for the FIFO and empty-flag read pointers. It is customary for it to align with the read-clock domain. Claiming the aempty_n signal is dependent on an augmentation of the rptr. Once the write pointer (wptr), synchronized with the rclk-domain instead of the read clock (rclk), increases, the signal is deasserted.

wptr_full.v

This module encompasses the complete flag logic and the FIFO write pointer, primarily operating within the write-clock domain. It is essential to increment the wrptr and ensure that the wrst_n transpires prior to asserting a full_n. A concurrent validation of the wclk domain and the input module signal, afull nsignal, is evident. In contrast to the wclk, the de-assertion of afull_nsignal occurs simultaneously with the increment of rptr. Filling and emptying occur at distinct moments.

Asynchronous Generation of Full And Empty

The async_cmp function exemplifies asynchronous processing by managing the aempty_n and afull_n signals. The asserted signal is the rising edge of a wclk, not a rclk. Conversely, an afull_n signal is activated and deactivated by a wclk event and a rclk event, respectively. Utilizing the vacant signal will terminate the reading session. To synchronize with the read clock, the leading and trailing edges of the empty signal must be in phase. A two-stage synchronizer facilitates the attainment of r_empty. The w_full signal is generated using the symmetrically equivalent method.

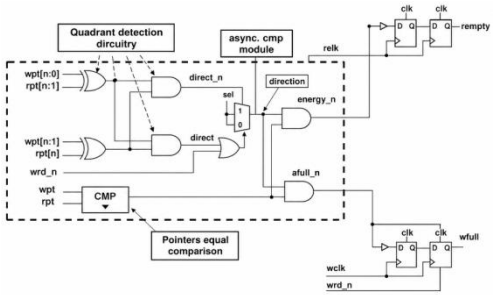


Fig-4 A comparison test is run asynchronously to evaluate whether the references are exhaustive and content-free.

Resetting the FIFO

- The First-In, First-Out (FIFO) mechanism is reset upon the occurrence of the initial significant FIFO event. Regarding the async_cmp module, the full and empty synchronizers of the wptr_full and rptr_empty modules, along with their interconnections, are regarded as observables. Upon resetting the First-In-First-Out (FIFO) system, these modules perform four essential functions.
- Immediately following the reception of the reset signal, the w_fullflag is eliminated. Resetting the device will not eliminate the r_emptyf delay.
- The pointer comparator assesses similarities subsequent to the reset signal clearing both FIFO pointers.
- The direction bit is eliminated during the reset operation.
- Once the direction bit is determined, the empty_n bit can be enabled; pre-setting the r_emptyflag renders both values identical.

Parallel-In, Parallel-Out, Universal Shift Register

The primary functions of the parallel-in/parallel-out shift register are to receive data in parallel format, shift it, and subsequently return it in parallel format. The universal shift register enables the simultaneous execution of several operations through its parallel-in/parallel-out capabilities.

The four data bits simultaneously inserted into the shift register are DA, DB, DC, and DD. The shift register can operate in either shifting mode or parallel loading mode by modifying the mode control, which accepts multiple inputs. Many technical devices may incorporate a mode control option that allows you to adjust your preferences as they evolve. With each clock pulse, a single bit of data is transmitted. The updated values can be obtained via the QC, QD, QA, and QB outputs.

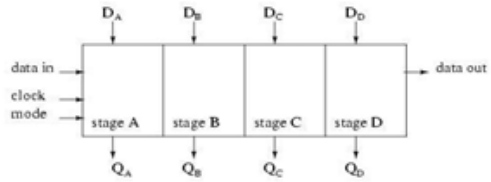


Fig-5:4- A stage shift register with parallel-in and parallel-out is a digital circuit component that can store and process binary data. It is notable for its capacity to transport data across phases and give input and output at the same time.

The terms "data in" and "data out" facilitate the progression of numerous phases. During this research, we will only disclose information pertaining to the most appropriate shifting technique. Two leftward-facing signals, data in and data out, can facilitate the lateral shift of the data flow. This schematic illustrates the components of a right-shifting, parallel-in/parallel-out shift register. Although tri-state buffers are not essential for the depicted parallel-in/parallel-out shift register, they have been incorporated anyway. Drawing inspiration from the concept of an optimal parallel-in/parallel-out shift register with right-shifting functionality, we utilize a condensed version of the information presented in the 74LS395 datasheet. The AND-OR multiplexer regulates data transfer to the FFs via the LD/SH signal. Four data inputs for flip-flops can be derived from the upper four AND gates (DA, DB, DC, and DD) when the LD/SH' ratio equals 1. Examine the clock inputs of the four flip-flops to verify the presence of the inverter bubble. As the negative edge of the shift signal transitions from low to high during a shift operation, this observation indicates that the 74LS395 is capable of storing data. In the subsequent clock cycle after synchronized synchronization, the four data bits (DA, DB, DC, and DD) will be transmitted to the output ports as QA, QB, QC, and QD, respectively. Data is accessible solely via the internal flip-flops and output pins when the actual segment of the system's output control (OC) is configured to a low state.

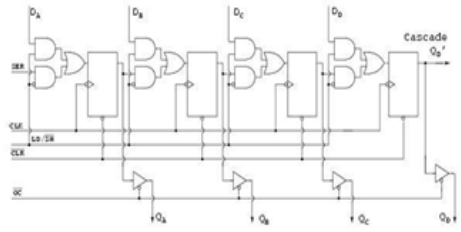


Fig-6: This talk investigates the alternating use of the tri-state mechanism by looking at parallelism in both internal and external situations.

The loaded data can be shifted one bit to the right if LD/SH' is zero on the subsequent negative clock edge. For example, consider the following. The information contained in our 4-bit shift register will be irretrievably lost after four clock cycles. The risk of data loss occurs when the device is not linked from its QD to the SER of another device.

In addition to the aforementioned Transfer Register, parallelism is a valuable concept with numerous potential uses.

	D_3	D_2	D_1	D_0
data	1	1	0	1
load	1	1	0	1
shift	x	1	1	0

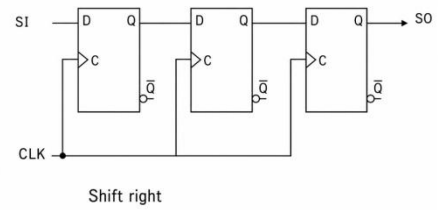
Load and shift

	D_3	D_2	D_1	D_0
data	1	1	0	1
load	1	1	0	1
shift	x	1	1	0
shift	x	x	1	1

Load and 2-shifts

Fig -7: This research looks at the data patterns of the sources DA, DB, DC, and DD..

Subsequently, it is forwarded to QA, QB, QC, and QD, where it undergoes a slight left shift. The observer is unaware of the nature of the X-designated data being received. A reference ground connection to the input signal (SER) may be utilized to ascertain the precise value of the transmitted data (0). Furthermore, it indicates that the rightward displacement measures two units, necessitating the use of two timepieces.



Shift right

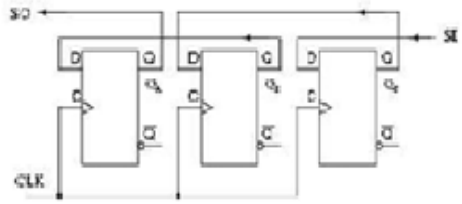
Fig-8: The term Shift Right refers to a computer operation that includes shifting the bits of a data structure to the right.

The subsequent graphic illustrates the hardware components utilized for transferring data to the right side of the screen. The primary objective is to highlight the simplicity of this graphic in comparison to others.

	Q_{in}	Q_B	Q_C
load	1	1	0
shift	X	1	1

Load and right shift.

The data has been displaced to the right in comparison to the preceding right-shifter above.



Shift left

Fig-9: Shift Left is a software development paradigm that prioritizes fault discovery early in the development process.

To execute a left shift, it is necessary to initially substitute the flip-flops (FFs). The most recent technological innovation surpasses its predecessor. The positioning of SI and SO is incorrect. Quebec, a province in Canada, will serve as the new domicile for the SI business. The sector is currently contemplating a transition from quality control (QC) to quality assurance (QA). Transitioning from quarterback to quality assurance analyst. The distinct item referred to as the shifter SI may be impacted when Quality Assurance (QA) loses connectivity with the source object (SO)..

	Q_A	Q_B	Q_C
load	1	1	0
shift	1	0	X

Load and left shift

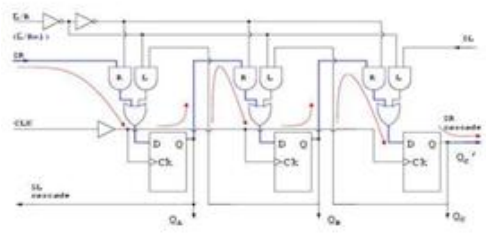


Fig-10: Choosing whether to go to the left or right and then carrying out the stated action

The hypothetical shift register can be manipulated in both directions with the letters L and R in the illustration. The conventional rightward direction is inverted when L'/R=1. When L divided by R equals one, the AND gates R engage the multiplexer.

The data is transmitted through the SR cascade output subsequent to traversing the QA, QB, and QC stages, following its entry into the SR input. Utilization of this pin may result in the lateral displacement of another entity's SR. What would occur if L'/R equaled zero?

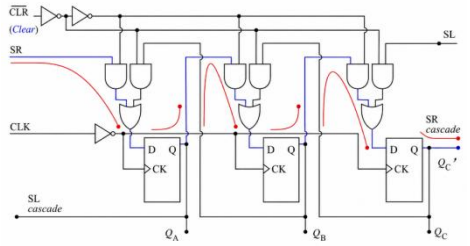


Fig-11: The cursor movement keys to the left and right, as well as the cursor movement key to the left.

When the L/R ratio reaches 0, the multiplexer and designated gates for L engage. The arrows in the previous leftward shift illustration indicate the identical trajectory established as a consequence of this. The procedure commences with data reception at the SL stage and proceeds through processing stages QC, QB, and QA. This pin can be utilized to shift another object's spatial position to the left. The previously referenced photos are visually compelling and elucidate the concept of shift left/right register. The reference to L'/R=0 in the left-right setup is explicitly elucidated. Concurrent data loading is essential for any enterprise component, as indicated by the section title. The accompanying illustration depicts the aforementioned occurrence. Having understood the functionality of L'/R gates for left and right shifting, we can now incorporate SH/LD', shift/load, and load AND gates into the system. This indicates that DC, DA, and DB can simultaneously serve as inputs. If SH/LD' is 0, then both the R and L gates are deactivated. The data can be transmitted from the DA port to the DB port, the DC port, and ultimately the FF port due to the presence of the BUT gates. Quality Assurance, Quality Control, and Quality Assurance will acquire the data in the subsequent CLK clock cycle.

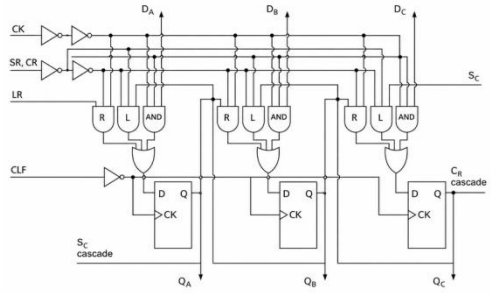


Fig-12: In the context of computer programming, loading and shifting is the process of transferring data in a predetermined direction, either to the left or to the right.

When SH/LD' equals 1, the load AND gates cease operation, and the L'/R control signals command the L AND gates to move left or right, accordingly. The graph before this one illustrates a modification to the ISAs. To function as an integrated device, the multiplexer necessitates a fourth AND gate, as previously indicated for the 74ALS299. The subsequent section illustrates this event specifically.

2.DESIGN AND ANALYSE ASYNCHRONOUS FIFO

Develop and evaluate a FIFO data structure employing an extensive range of read and write operations. Each of the sixty-four reviewed sources had thirty-two items of information. The First-In-First-Out (FIFO) method for organizing computer work requests prioritizes processing for the request that was received first from a stack or queue. Data may be retained in a singular clock domain and subsequently transmitted to alternate clock domains as required, utilizing read/write memory and flip-flops with First-In, First-Out (FIFO) logic. The clock domains tasked with transmitting data to and receiving data from the FIFO are referred to as the write logic and read logic, respectively.

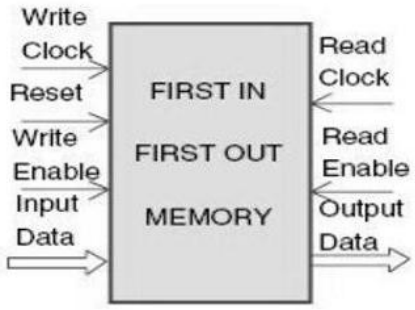


Fig-13: FIFO (First-In, First-Out) memory input/output (I/O).

3.CONCLUSIONS

The architecture of pointers and the distinction between full and empty states must be meticulously evaluated while creating an asynchronous First-In-First-Out (FIFO) system. When critical components are overlooked, the design is frequently defective and readily demonstrable. Identifying design flaws in First-In-First-Out (FIFO) architectures is a prevalent function of gate-level simulators. These models gather and monitor actual output delays. Gray code pointers facilitate the secure synchronization of FIFO pointers between clock domains. To provide precise synchronization of FIFO pointers across many clock domains, gray code pointers may be employed. Locating the entire state might be exceedingly cumbersome when attempting to implement a FIFO technique. An n-bit pointer can synchronize with an alternate clock domain, whereas a (n-1)-bit pointer facilitates comprehensive comparisons of dual n-bit Gray code counters. Current techniques for synchronizing binary FIFO pointers are more effective within the context of FIFO design. The FIFO-empty state can be ascertained by comparing and equalizing the synchronized n-bit write pointer and the n-bit read pointer. The paper's methodologies aim to resolve problems associated with asynchronous timers that exhibit varying degrees of conflict.

REFERENCES

[1] N. Verma, “Analysis Towards Minimization of Total SRAM Energy over Active and Idle Operating Modes,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 9, pp. 1695–1703, Sept. 2010.

[2] [2] K. Nii et al., “A 65 nm Ultra-High-Density Dual-Port SRAM with 0.71 µm² 8T Cell for SoC,” Proceedings of the IEEE Symposium on VLSI Circuits, pp. 130–131, 2006.

[3] [3] W.-H. Du et al., “An Energy-Efficient 10T SRAM-Based FIFO Memory Operating in Near-/Sub-Threshold Regions,” Proceedings of the IEEE System-on-Chip Conference, pp. 19–23, Sept. 2011.

[4] [4] D. Marković et al., “Ultralow-Power Design in Near-Threshold Region,” Proceedings of the IEEE, vol. 98, no. 2, pp. 237–252, Feb. 2010.

[5] [5] I.-J. Chang et al., “A 32 kb 10T Sub-Threshold SRAM Array with Bit-Interleaving and Differential Read Scheme in 90 nm CMOS,” IEEE Journal of Solid-State Circuits, vol. 44, no. 2, pp. 650–658, Feb. 2009.

[6] [6] Y.-T. Chiu et al., “Subthreshold Asynchronous FIFO Memory for Wireless Body Area Networks (WBANs),” International Symposium on Medical Information and Communication Technology (ISMICT), Mar. 2010.

[7] [7] M.-H. Tu et al., “Single-Ended Subthreshold SRAM with Asymmetrical Write/Read-Assist,” IEEE Transactions on Circuits and Systems, vol. 57, no. 12, pp. 3039–3047, Dec. 2010.

[8] [8] M.-T. Chang et al., “A Robust Ultra-Low-Power Asynchronous FIFO Memory with Self-Adaptive Power Control,” Proceedings of the IEEE System-on-Chip Conference, pp. 175–178, 2008.

[9] [9] W.-H. Du et al., “A 2 kb Built-in Row-Controlled Dynamic Voltage Scaling Near-/Sub-Threshold FIFO Memory for WBANs,” Proceedings of the IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT), pp. 1–4, 2012.

