Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 03 Issue: 11 | Nov-2025
ISSN (O) 3107-6696

# Synchronous V/S Asynchronous FIFO Design

**Dr. Shailaja Mudengudi[1], Komal L Mane[2], Laxmi B Amargol[3], Vani S Sarvi[4]**

*Department of Electronics and Communication Engineering Visvesvaraya Technological University, Belagavi Gadag, Karnataka, India*

*Komalm0851@gmail.com[2], laxmiamargol@gmail.com[3], vanissarvi@gmail.com[4]*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract -** This project presents the design, implementation, and comparative analysis of Synchronous FIFO and Asynchronous FIFO architectures using Verilog HDL and EDA tools. FIFO (First-In-First-Out) memories are widely used in digital systems for temporary data storage and reliable communication between subsystems. However, designing FIFO architectures that ensure high speed, low power, and reliable clock-domain crossing remains challenging.

In this work, both FIFO types are designed and analyzed under identical conditions.
The Synchronous FIFO uses a single clock domain, enabling simple control logic and high throughput, while the Asynchronous FIFO uses dual clock domains and Gray code–based pointer synchronization to avoid metastability. Functional verification, synthesis, and power analysis are performed using RTL simulations and synthesis tools. The results show that synchronous FIFOs achieve higher speed and simpler design, whereas asynchronous FIFOs provide reliable data transfer between different clock domains. The final comparison highlights the trade-offs between complexity, reliability, speed, and power.

***Keywords***: Synchronous FIFO, Asynchronous FIFO, Clock Domain Crossing, Gray Code Pointers, Verilog HDL, RTL Design, FIFO Architecture, Metastability, VLSI, Digital System Design.

## Introduction

FIFO (First-In-First-Out) memory buffers are essential building blocks in modern digital systems. They are used in communication interfaces, processors, SoCs, network devices, and real-time data streaming applications. Their main purpose is to temporarily hold data and ensure ordered transmission from the producer to the consumer.

A Synchronous FIFO operates using a single clock signal. All write and read operations occur on the same clock edge, resulting in simple logic, easy timing closure, and predictable performance. These FIFOs are typically used in systems where both sender and receiver run at the same frequency.

In contrast, an Asynchronous FIFO is used when the write and read operations occur in different clock domains. Because of this clock mismatch, metastability issues arise. Thus, Gray-coded read and write pointers along with synchronizer flip-flops are used to ensure reliable operation.

With increasing SoC complexity and the heavy use of multiple processing blocks, choosing the correct FIFO architecture in terms of speed, power, and reliability has become essential.

This project focuses on:

- Designing both FIFO architectures using Verilog HDL
- Performing functional verification
- Analyzing timing, area, and power through synthesis
- Comparing both architectures in terms of efficiency and reliability

This work demonstrates how architectural choices impact the overall performance of FIFO-based data communication systems.

## II.RELATED WORK

FIFO architecture design has been studied extensively, especially in areas involving clock domain crossing (CDC) and low-power memory structures.

Several works highlight reliability issues in asynchronous FIFOs due to metastability, which must be resolved using techniques such as pointer synchronization and Gray-code addressing. Research by Sun et al. emphasized that asynchronous FIFOs require careful pointer synchronization using dual-flip-flop synchronizers to avoid incorrect flag generation.

High-speed synchronous FIFO designs often rely on pipelining, optimized pointer arithmetic, and reduced control complexity. Prior works show that synchronous FIFOs achieve lower latency since both read and write occur on the same clock.

Studies on low-power FIFO designs focus on minimizing switching activity, optimizing memory read/write logic, and reducing unnecessary transitions in pointer generation.

Researchers have also explored dynamic frequency scaling and gated clock techniques for power reduction.

However, few comparative studies exist where synchronous and asynchronous FIFOs are analyzed under identical conditions, including RTL design, synthesis, timing, and power analysis. This project fills that gap by providing a unified framework and practical implementation comparison.

## III. ARCHITECTURE OVERVIEW

The architecture of a FIFO consists of a memory array, read pointer, write pointer, and control logic that manages the data flow. Although both synchronous and asynchronous FIFOs use the same basic components, their internal operation differs depending on the clocking method.

A Synchronous FIFO uses a single clock for both read and write operations. Because of this shared clock, the pointer updates, memory access, and full/empty flag generation happen in the same timing domain. The read and write pointers are simple binary counters, and the comparison between them is straightforward.

Since no clock domain crossing exists, there is no risk of metastability, making the design simpler, faster, and easier to implement.

In contrast, an Asynchronous FIFO is designed for systems where the write and read sides operate on entirely different clocks. This difference in clocks introduces timing uncertainty, so the architecture uses Gray-coded pointers instead of binary, ensuring that only one bit changes at each increment. These Gray- coded pointers are safely transferred across clock domains using synchronizer flip-flops. Once synchronized, they are compared to generate full and empty conditions. Although this architecture is more complex due to clock-domain crossing, it reliably transfers data between unrelated clock frequencies.

Overall, both FIFO types use the same fundamental structure, but synchronous FIFO focuses on speed and simplicity, whereas asynchronous FIFO emphasizes safe and reliable data transfer between independent clock domains.

## IV. METHODOLOGY

The design flow used in this project follows standard RTL digital design methodology.

### 4.1. Specification and Design Requirements

The project begins by defining FIFO depth, data width, expected throughput, and power constraints. These specifications help determine the size of the memory array and pointer width. For asynchronous FIFO, the identification of clock frequencies is essential because the difference in clock rates dictates synchronization requirements and affects metastability probability.

During specification, it is also important to define acceptable latency, maximum operating frequency, and flag behavior.

For example, synchronous FIFOs may operate at very high clock speeds without requiring additional safety mechanisms, whereas asynchronous FIFOs must consider metastability windows and safe synchronization periods.

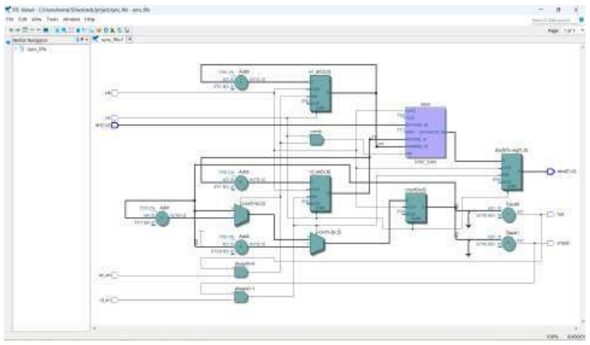
### 4.2 RTL Design Using Verilog HDL

Both FIFO types were modeled using Verilog HDL in a modular, hierarchical manner. The memory module stores data, while the pointer modules manage addressing. For synchronous FIFO, pointer arithmetic and full/empty detection were implemented using simple binary comparisons. In asynchronous FIFO, additional modules were developed to convert binary pointers to Gray code and vice versa. Synchronizer modules, consisting of chains of flip-flops, were added to safely transfer Gray-coded pointers across clock domains.

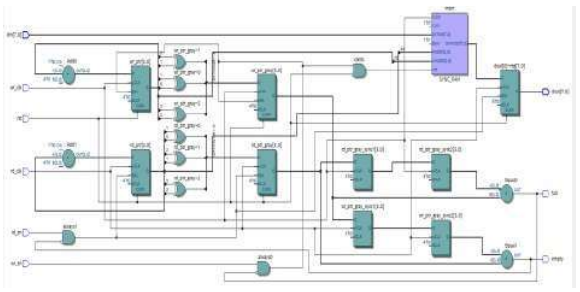
Designing the control unit required careful treatment of boundary cases such as pointer wrap-around, simultaneous read/write conditions, and FIFO initialization after reset. The RTL code was structured to be synthesizable, hardware-efficient, and easy to debug

Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 03 Issue: 11 | Nov-2025
ISSN (O) 3107-6696

#### 4.2.1. RTL Design of Synchronous FIFO



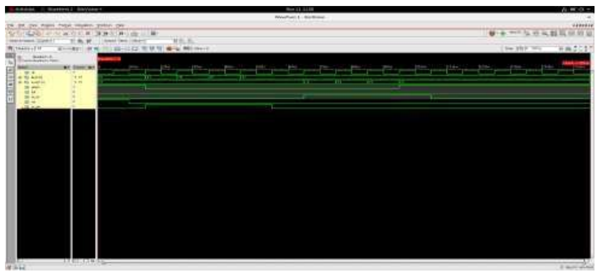
#### 4.2.2 RTL Design of Synchronous FIFO



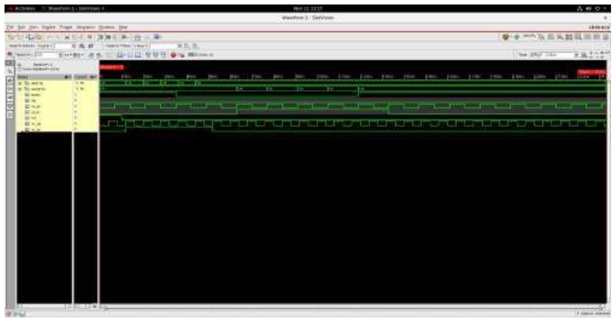
### 4.3 Functional Simulation

Simulation was carried out to validate the correctness of FIFO behavior. For synchronous FIFO, tests included performing fast consecutive writes, burst reads, and simultaneous operations to observe pointer transitions. For asynchronous FIFO, simulation was more complex because the read and write clocks were set to different frequencies and phases to replicate real-world asynchronous behavior. The simulation environment tested metastability scenarios, pointer synchronization timing, and corner cases such as slow-write fast-read patterns. All expected read outputs were compared against reference models to ensure no data corruption, pointer misalignment, or flag generation errors occurred.

#### 4.3.1. Simulation Result of Synchronous FIFO Design



#### *4.3.1.* Simulation Result of Asynchronous FIFO Design

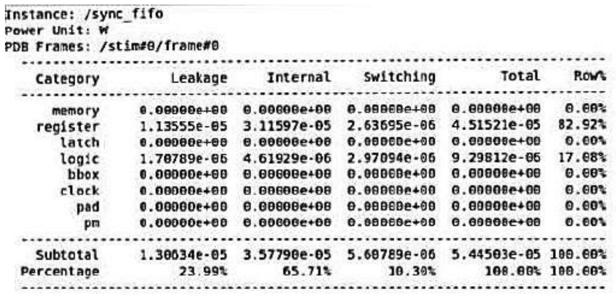


### 4.4 Synthesis Using Cadence Genus

The RTL descriptions were synthesized using standard-cell technology to estimate hardware resource utilization, maximum operating frequency, and power consumption. The synthesis tool converted HDL code into gate-level netlists, and timing constraints were applied to achieve the required performance targets.

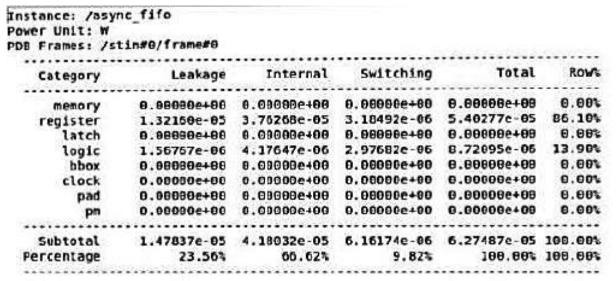
Synchronous FIFOs generally showed superior timing results due to their single-clock nature and minimal control complexity. Asynchronous FIFOs consumed slightly more area due to additional synchronizers and Gray code logic

### 4.5 Final Result - Power of Synchronous FIFO



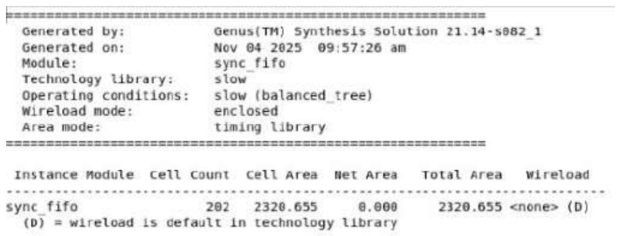
```
Instance: /sync_fifo
Power Unit: W
PDB Frames: /stim#0/frame#0
  --------------------------------------------------------------------------
     Category         Leakage     Internal    Switching        Total    Row%
  --------------------------------------------------------------------------
       memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
     register     1.13555e-05  3.11597e-05  2.63695e-06  4.51521e-05  82.92%
        latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        logic     1.70789e-06  4.61929e-06  2.97094e-06  9.29812e-06  17.08%
         bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        clock     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
          pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
           pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  --------------------------------------------------------------------------
     Subtotal     1.30634e-05  3.57790e-05  5.60789e-06  5.44503e-05 100.00%
   Percentage          23.99%       65.71%       10.30%      100.00% 100.00%
  --------------------------------------------------------------------------
```

#### Power of Asynchronous FIFO



```
Instance: /async_fifo
Power Unit: W
PDB Frames: /stim#0/frame#0
  --------------------------------------------------------------------------
     Category         Leakage     Internal    Switching        Total    Row%
  --------------------------------------------------------------------------
       memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
     register     1.32160e-05  3.76268e-05  3.10492e-06  5.40277e-05  86.10%
        latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        logic     1.56767e-06  4.17647e-06  2.97682e-06  8.72095e-06  13.90%
         bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        clock     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
          pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
           pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  --------------------------------------------------------------------------
     Subtotal     1.47837e-05  4.18032e-05  6.16174e-06  6.27487e-05 100.00%
   Percentage          23.56%       66.62%        9.82%      100.00% 100.00%
  --------------------------------------------------------------------------
```

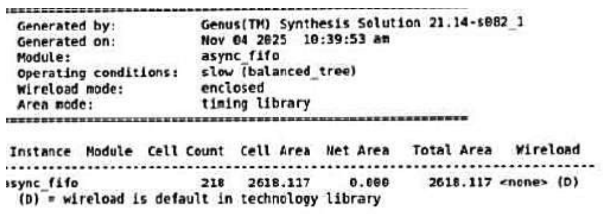
#### Area of Synchronous FIFO



```
============================================================
  Generated by:            Genus(TM) Synthesis Solution 21.14-s082_1
  Generated on:            Nov 04 2025  09:57:26 am
  Module:                  sync_fifo
  Technology library:      slow
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

 Instance Module  Cell Count  Cell Area  Net Area   Total Area   Wireload
--------------------------------------------------------------------------
sync_fifo                202   2320.655     0.000     2320.655 <none> (D)
  (D) = wireload is default in technology library
```

**Area of Asynchronous FIFO**



```
============================================================
  Generated by:            Genus(TM) Synthesis Solution 21.14-s082_1
  Generated on:            Nov 04 2025  10:39:53 am
  Module:                  async_fifo
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

 Instance  Module  Cell Count  Cell Area  Net Area   Total Area   Wireload
--------------------------------------------------------------------------
async_fifo               218   2618.117     0.000     2618.117 <none> (D)
  (D) = wireload is default in technology library
```

### 4.6 Design Iteration and Optimization

If timing constraints were violated or if power consumption exceeded acceptable limits, modifications were made at the RTL level. This included optimizing pointer logic, reducing redundant transitions, and simplifying control circuitry. For asynchronous FIFOs, synchronizer stages were examined closely to ensure reliability without excessive delay.

## V. REFERENCES

- C. E. Cummings, "Simulation and Synthesis Techniques for Asynchronous FIFO Design with Gray Code Pointer," Synopsys Users Group (SNUG), San Jose, 2002.
- K. E. Lochner, "Metastability in Clock Domain Crossings," IEEE Design & Test of Computers, vol. 28, no. 5, pp. 36–47, 2011.
- J. Liu and D. Thomas, "Design of a High-Performance Synchronous FIFO," IEEE International Conference on Computer Design (ICCD), pp. 150–155, 2009.
- Xilinx Inc., "FIFO Generator v13.2—Product Guide," Xilinx Documentation, 2018
- K. S. Miskov-Zivanov and D. Marculescu, "Modeling and Analysis of Uncertainty in Clock Domain Crossing Circuits," IEEE Transactions on Computer-Aided Design, vol. 29, no. 10, pp. 1597–1608, 2010
- A. Singh and M. Singh, "A High-Speed Asynchronous FIFO for Reliable Data Transfer," IEEE International SOC Conference, pp. 225-228-2014.
- J. Sparsø and S. Furber, Principles of Asynchronous Circuit Design, Springer, 2020
- M. Greenstreet, "Understanding and Avoiding Metastability in Digital Systems," IEEE International Symposium on Asynchronous Circuits and Systems, pp. 164–175, 2015
- R. Nair, "Clock Domain Crossing: Techniques and Timing Challenges," IEEE Microelectronics Journal, vol. 44, no. 2, pp. 112–123, 2017
- P. Day and G. Woods, "Investigation into FIFO Architectures for High-Speed Communication Systems," IEEE Transactions on Circuits and Systems, vol. 49, no. 3, pp. 421–432, 2002.