2025 IEEE 17th International Conference on Computational Intelligence and Communication Networks (CICN) | 979-8-3315-8733-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/CICN67655.2025.11368188

Unified Power Format Driven Low Power Asynchronous FIFO Design for CDC Applications

Nikhil
School of ECE
REVA University
Bengaluru, India
nikhilpechatti97@gmail.com

Prameela Kumari N
School of ECE
REVA University
Bengaluru, India
prameela.n@reva.edu.in

***Abstract*—Asynchronous FIFOs are commonly used to manage Clock Domain Crossing (CDC) in digital systems, but conventional designs consume excessive power due to continuous operation and lack of power management features. This paper presents a Unified Power Format (UPF)-based methodology for low-power Register-Transfer Level (RTL) design of an asynchronous FIFO using Xilinx Vivado. The proposed UPF-enabled architecture introduces retention registers and isolation logic across distinct power domains to enable controlled switching, state retention and low-power operation. During power-down, outputs are clamped and pointer states are saved in retention registers. Upon power restoration, pointer states are reloaded, enabling reliable synchronization and correct operation across clock domains. Simulation results confirm functional correctness under both normal and power-aware conditions, ensuring signal integrity and preserved FIFO state. Power estimation indicates that the proposed UPF-enabled design achieves 92% overall reduction in total power compared to a conventional FIFO without UPF integration. Overall, this hybrid RTL–UPF approach provides an energy-efficient solution for CDC design, suitable for both FPGA and ASIC implementations.**

***Index Terms*—Asynchronous FIFO, Clock Domain Crossing (CDC), Gray Coded Pointers, Synchronization, Unified Power Format (UPF).**

I. INTRODUCTION

Clock domain crossing (CDC) is a common challenge in modern digital systems, as it involves transferring data between modules operating under different clocks. Improper synchronization during CDC can lead to metastability, data corruption or loss, effects system reliability. Asynchronous First-In-First-Out (FIFO) buffers are widely used to enable safe data transfer across clock domains [1][2].

Asynchronous FIFOs are critical in bridging these clock domains by buffering data and managing the full/empty flags [3]. The risks of metastability during CDC will always be a problem even in pointer and flag-based logic. This issue is typically mitigated through Gray-coded pointers combined with dual-stage synchronizers [4]. While this design guarantees functional correctness, but it does not address power consumption.

UPF is extensively used in ASIC design flows, but FPGA design environments such as Xilinx Vivado primarily provide power estimations from synthesized netlists. Despite these limitations, UPF methodologies can still be integrated into FPGA prototyping, particularly for asynchronous FIFOs. By retaining Gray-coded pointers during power-down sequences, these FIFOs avoid the need to rewrite data upon resuming power, maintaining functional integrity while enabling power optimization.

II. BACKGROUND

A. CDC and FIFO Fundamentals

Asynchronous FIFOs are a widely adopted solution for CDC [5]. They consist of independent read and write pointers operating under different clocks. Gray-coded pointers are employed to ensure that only a single bit changes at a time, minimizing the risk of metastability. Additionally, dual flip-flop synchronizers are applied to pointer values when crossing domains. While this structure ensures functional correctness in CDC scenarios, it does not inherently address power optimization, which is critical in both FPGA and ASIC designs [6][7].

B. Unified Power Format (UPF)

The Unified Power Format (UPF) is an IEEE 1801 standard that allows designers to specify power intent independently of RTL functionality [8][9]. UPF is widely used in ASIC design flows but has limited support in FPGA environments like Xilinx Vivado. Integrating UPF methodologies into FPGA design flows allows to estimate power from netlist generated.

UPF also allows specifying Power domains, Power switches, Isolation strategies and Retention strategies without affecting the functional behavior of the design [10].

C. UPF in FPGA Prototyping and CDC FIFOs

Although FPGAs are primarily used for functional prototyping, applying UPF-based methodologies can provide significant benefits in low-power design validation. In the context of asynchronous FIFOs, UPF facilitates power-intent simulation, including the effects of isolation and retention. Retention registers hold read and write pointer values during power-down events, while isolation logic ensures safe clamping of read and write outputs. These RTL-level modifications emulate ASIC-like power behavior on FPGA platforms.

979-8-3315-8733-8/25/$31.00 ©2025 IEEE
DOI: 10.1109/CICN.2025.209

By integrating retention and isolation logic, designers can compare power consumption between baseline and low-power architectures, enabling early evaluation of energy-efficient design strategies. This approach allows FPGA prototypes to not only verify functional correctness but also assess power-aware design techniques, bridging the gap between FPGA prototyping and ASIC low-power design flows.

III. Proposed Asynchronous FIFO Architecture

The asynchronous FIFO with UPF is designed to provide reliable communication between independent clock domains while minimizing dynamic power consumption through power gating. The overall methodology integrates standard FIFO design principles with power intent described in UPF. At a functional level, the FIFO enables input data to be written in the write clock (wclk) domain and read in the read clock (rclk) domain, with memory pointers and status flags ensuring correct operation. At the power level, additional control signals manage domain shut-down, output isolation and pointer retention so that the FIFO state is preserved during low-power modes and restored on wake-up.

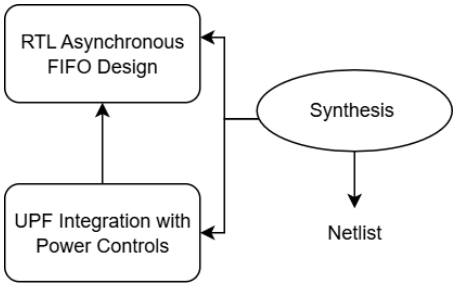

Fig.1 Block Diagram of UPF Enabled Asynchronous FIFO Design

Fig.1, illustrates the block diagram of the UPF-enabled Asynchronous FIFO design. The RTL-FIFO design is integrated with UPF-based power control definitions before synthesis. The synthesis tool utilizes both the RTL and UPF descriptions to generate a power-aware netlist, which incorporates isolation cells, retention registers and power-switching logic. This power-aware netlist is then used for simulation and implementation in Xilinx Vivado to evaluate functionality and power performance.

A. RTL and UPF-Aware Design Flow

The design flow combines register-transfer level (RTL) modeling with Unified Power Format (UPF) specifications to achieve low-power functionality.

The RTL captures the functional behavior of the FIFO, including pointer updates, memory access and synchronizer logic, while the UPF provides power intent definitions for sleep, isolation and retention strategies. Together, these ensure that the FIFO performs correctly under normal operation and maintains functional integrity during power gating. The overall operational design flow is illustrated in Fig.2, including sub flows.

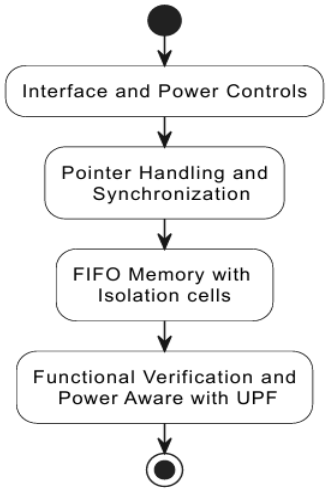

Fig.2 Operational flow of design

B. Interface and Power Controls

The FIFO operates across two asynchronous domains, controlled by independent write (wclk) and read (rclk) clocks. Write operations are driven by w_en, wrst_n, and data_in, while read operations use r_en, rrst_n, and data_out. In addition to these, UPF signals are integrated to manage low-power operation. The fifo_sleep signal disables the FIFO domain when idle, and the fifo_iso_en signal isolates the outputs to prevent propagation of undefined values into the always-on domain. Before power-down, the current values of the write and read pointers are stored in retention registers (saved_wptr, saved_rptr). On wake-up, these retained values are restored into active pointers (load_wptr_val, load_rptr_val), ensuring that the FIFO resumes correct operation without loss of data integrity.

C. Pointer Handling and Synchronization

After the interface and power control logic, the next stage in the design flow ensures that write and read pointers are handled reliably across asynchronous domains. as illustrated in Fig.3, the process begins by checking the enable condition along with the status flag.

When the condition is satisfied, the binary pointer (b_ptr) is incremented by one and its Gray-coded equivalent (g_ptr_next) is generated using a binary-to-gray conversion. Both the binary and gray pointers are then updated to reflect the new position. If the condition is not satisfied, the pointers remain unchanged, thereby preventing invalid updates.

The updated gray pointer is compared against the synchronized pointer from the opposite domain (g_ptr_sync) to determine the status flag, such as full or empty. This ensures that no overflow or underflow occurs during operation, which avoids metastability and dedicated synchronizers transfer gray-coded write pointers across domains.

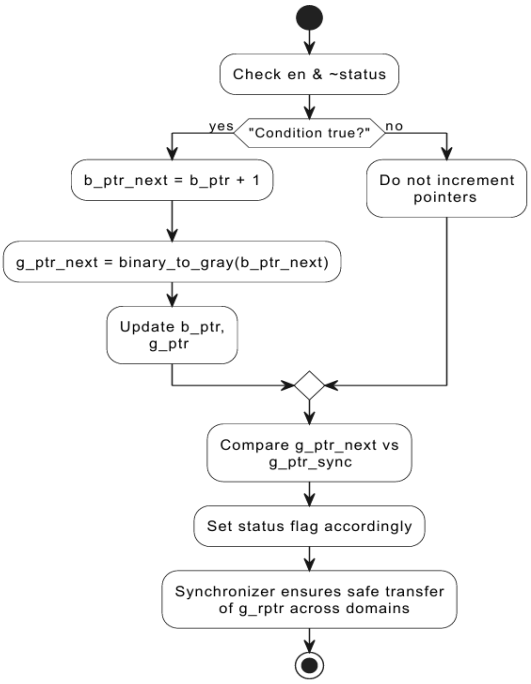

Fig.3 Flow of pointer handling and synchronizer

D. FIFO Memory with Isolation Cells

The FIFO memory block serves as the shared storage interface between the write and read clock domains. Its primary role is to store incoming data from the write domain and provide it to the read domain while maintaining proper synchronization across independent clocks.

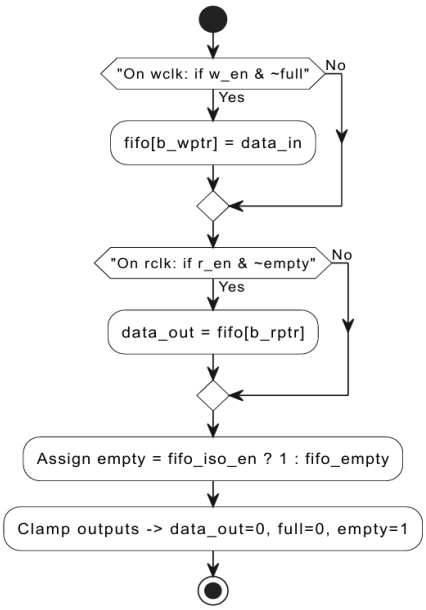

Fig.4 Flow of FIFO memory

As illustrated in Fig.4 FIFO memory access follows a dual-clock operation. In the write domain, valid data is written to the memory when w_en is asserted and the full flag is low. The write pointer is incremented and its Gray-coded equivalent is updated. Similarly, in the read domain, data is retrieved when r_en is asserted and the FIFO is not empty, with the read pointer being updated accordingly. Synchronization ensures that pointers are consistently transferred across domains, enabling correct flag generation.

In normal operation (isolation disabled), the FIFO behaves as a conventional dual-port memory, allowing independent read and write operations without data corruption. The design ensures seamless clock domain crossing while maintaining data integrity and system stability.

E. Power Aware with UPF

With UPF integration, the FIFO design is extended to operate reliably under power-gated conditions as shown in Fig.5. The pointer handling mechanism is made power-aware by retention logic which secures the current pointer state before the domain is powered down. When fifo_sleep is asserted, the write and read pointers are captured into retention registers (saved_wptr, saved_rptr) and the outputs are isolated through fifo_iso_en to prevent propagation of undefined values. Upon reactivation, the retained values (load_wptr_val, load_rptr_val) are restored, allowing the FIFO to resume operation seamlessly from its previous state without corruption. This ensures that the continuity of data handling is preserved even across power cycles.

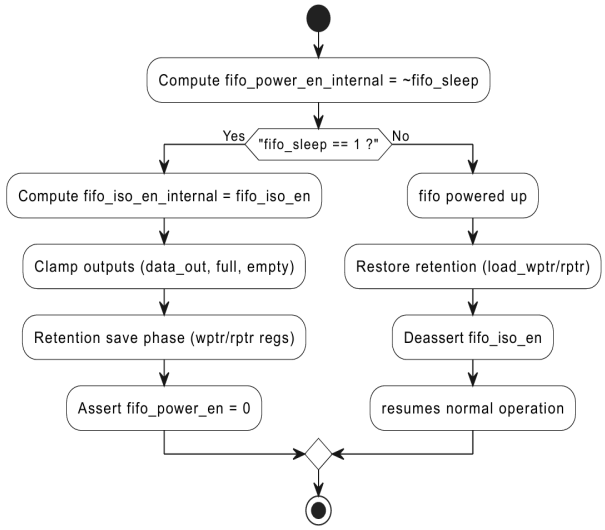

Fig.5 Flow of power aware with UPF

During normal operation, the process begins when the power enable (power_en) signal activates the write logic, permitting data_in to be stored at the memory location indicated by the write pointer. The retained and restored pointer values are then used to correctly evaluate the full condition and generate the full flag, preventing overflow. On the read side, data is accessed as data_out.

Verification of the design was carried out at both functional and power-aware levels. At the functional level, signals such as debug_b_wptr, debug_b_rptr and

write_index were monitored to confirm correct pointer increments, memory addressing and flag generation.

At the power-aware level, UPF simulations validated the correct operation of power switches, isolation cells and retention registers. When the FIFO entered low-power mode (fifo_sleep = 1), the power domain was shut down, outputs were clamped and pointer states were saved. On wake-up, power was restored, isolation was released and the retained pointers were reloaded, enabling the FIFO to continue operation without data loss.

IV. Result and Analysis

A. Functional Verification

The functionality of the asynchronous FIFO was first verified without UPF integration to establish baseline correctness. As shown in Fig.6, the simulation waveform confirms that input data (data_in) was successfully written into the FIFO whenever the write enable signal was asserted. Data words were written sequentially at different time intervals. During the read phase, the stored data was retrieved in the same order, with the output sequence (data_out) matching the input sequence (data_in), verifying correct FIFO operation. The status flags (full, empty) were accurately generated and signals such as write_index[31:0] confirmed proper pointer incrementing and memory indexing. These observations validate the functional correctness of the asynchronous FIFO under normal operating conditions.

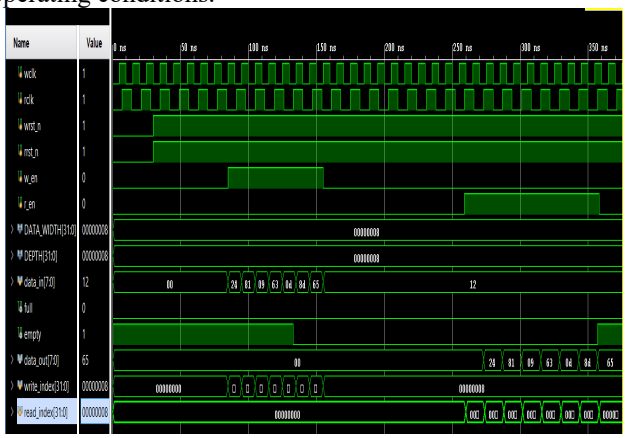

Fig.6 Waveform of Asynchronous FIFO without UPF

After integrating Unified Power Format (UPF), additional signals such as power_en and iso_en were introduced to enable power-aware verification. The corresponding simulation waveform, shown in Fig.7, illustrates the FIFO's behavior during power transitions. Before power-down, the write and read pointer states were stored in retention registers (saved_wptr = 7, saved_rptr = 0), ensuring preservation of internal states. During the power-off interval, debug signals (debug_b_wptr = 7, debug_b_rptr = 0) remained stable, confirming that pointer values were retained while output isolation prevented undefined signal propagation.

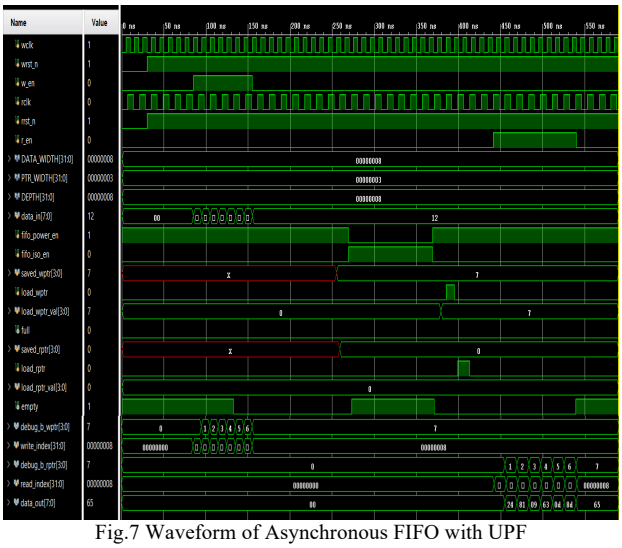

Fig.7 Waveform of Asynchronous FIFO with UPF

Upon power restoration, the retained pointer states were reloaded, allowing seamless resumption of operation. The output data sequence and pointer progression continued correctly, confirming state retention, synchronization integrity and functional continuity across power cycles.

These results demonstrate that the proposed UPF-enabled asynchronous FIFO operates correctly under both normal and power-aware conditions, maintaining reliable data transfer and synchronization across clock domains without loss or corruption.

B. Power Analysis

Power estimation was carried out for both conventional and UPF-enabled asynchronous FIFO designs using the Xilinx Vivado power analysis environment.

TABLE I. Comparison of Power with and without UPF

Component	Without UPF(W)	With UPF(W)
Dynamic Power	1.824	0.007
Static Power	0.136	0.131
Logic	0.145	<0.001
RAM	0.020	<0.001
Register	0.019	<0.001
Signals	0.304	<0.001
I/O	1.325	0.005
Total Power	**1.960**	**0.138**

In the baseline FIFO (without UPF), the total on-chip power consumption was measured at 1.960 W, consisting of 1.824 W dynamic power and 0.136 W static power. The significant amount of dynamic power comes not only from any clock operation present in the logic, registers and I/O buffers, but from static clock activity even during situations where no read/write operations were being performed. Overall, the dynamic energy component is high since all components remained powered and clock switching was present, even in idle periods.

With UPF integration, the same design exhibited a total power consumption of only 0.138 W, with dynamic power reduced to 0.007 W and static power at 0.131 W, as summarized in Table I. This corresponds to a 99% reduction in dynamic power and a 92% reduction in total power compared to the conventional design.

V. Conclusion

This work presents the design, implementation and power aware for asynchronous FIFO with a hybrid RTL UPF methodology. While the FIFO design was functionally correct, it had high dynamic power due to continuous switching activity in the logic, signals and I/O paths. With UPF integration, the design supports low-power operation without losing the integrity of the data.

Simulation results verified that pointer states during the power-down mode retained the exact states and restored correctly on power-up mode in the FIFO continued operation. Power analysis showed 92% reduction in total compared to FIFO without UPF. This methodology is extensible to other CDC based systems and it is energy-efficient for FPGA and ASIC designs.

Acknowledgment

The authors would like to thank the REVA University, Bengaluru, for providing the necessary support and resources to complete this work.

References

[1] Ying Xu, “Asynchronous FIFO design based on Verilog,” Journal of Physics: Conference Series, 2022.

[2] Mingyu Liu, Jing Yang, and Zhiwei Xu, “Design of asynchronous FIFO based on Verilog HDL,” in IOP Conf. Series: Materials Science and Engineering, vol. 677, 2019.

[3] E. Xie, J. Zhou, “Analysis and Comparison of Asynchronous FIFO and Synchronous FIFO,” in IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), 2023.

[4] Clifford E. Cummings, “Simulation and Synthesis Techniques for Asynchronous FIFO Design,” in SNUG San Jose, 2002.

[5] V. Patel, V. Vimal, J. Patoliya, B. Soni, “Design & Implementation of Novel Asynchronous FIFO,” in IEEE International Symposium on Smart Electronic Systems (iSES), 2023.

[6] X. Ren, C. Li, K. Liu, “Design and Implementation of High Reliability FIFO Based on FPGA,” in IEEE Information Technology and Mechatronics Engineering Conference (ITOEC), 2023.

[7] S. Kalyan, A. Porel, A. Chandra, “Power, Performance and Area Optimization ofAsynchronous FIFO,” in IEEE International Symposium on Smart Electronic Systems (iSES), 2024.

[8] IEEE Std 1801-2018, IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems, IEEE, 2018.

[9] IEEE Std 1801-2024, IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems, IEEE, 2024.

[10] B. Pandey, M. Rajina, GS Tomar, DM Akbar, EB Yousef, “FSM Based Green Memory Design and its Implementation on Ultrascale Plus FPGA”, Journal of Critical Reviews, Vol.7(19), pp.454-458, 2020.

[11] Avinash Yadlapati and K. Hari Kishore, “Low power synthesis for asynchronous FIFO using UPF,” International Journal of Innovative Technology and Exploring Engineering, 2018.