VERIFICATION TEST PLAN

(Asynchronous FIFO - Functional Verification)

ECE-593: Fundamentals of Pre-Silicon Validation
Maseeh College of Engineering and Computer Science
Winter, 2025

Project Name: Design, Implementation and Verification of Asynchronous FIFO

Members:

Sai Charan Teja Balne

Akshay Kumar Medishetti

Steeva Murikipudi

Date:

Table of Contents

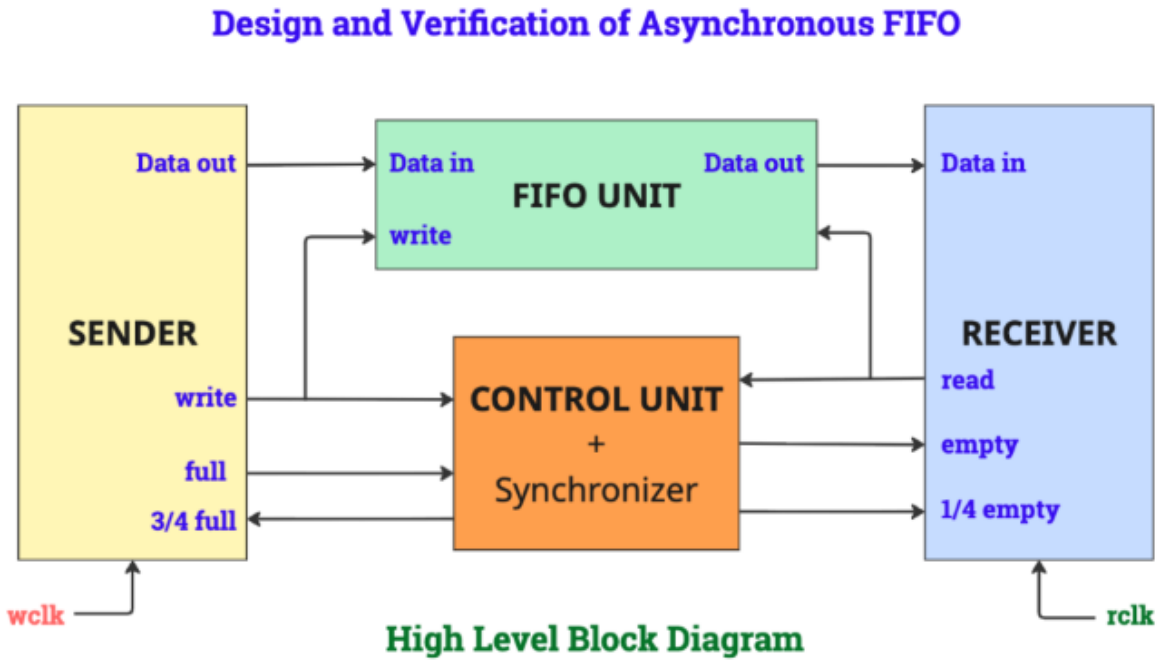
Introduction:

Objective of the verification plan

The objective of this verification plan is to validate the design and implementation of an asynchronous FIFO buffer that enables safe data transfer between two independent clock domains. This plan describes a multi-stage verification approach, starting with basic functional checks and progressing to an advanced UVM-based testbench. The main goal is to achieve 100% code coverage and functional coverage by systematically verifying the following aspects:

- Basic FIFO functionality, including write, read, full, and empty operations
- Correctness of clock domain crossing (CDC) mechanisms
- Proper generation and timing of status flags
- Data integrity under different operating conditions
- Boundary conditions and corner cases

Top Level block diagram



Specifications for the design

Sender Clock Frequency: 500 MHz

Write Idle Cycles: 0

Write Burst Size: 450 items
Receiver Clock Frequency: 500 MHz
Read Idle Cycles: 4 cycles per read
There are no idle cycles between two successive write operations. This means that one data item is written in every clock cycle.
There are four idle clock cycles between two successive read operations. That is, after reading one data item, Module B waits for four clock cycles before initiating the next read. Therefore, one data item is read every five clock cycles.
The time required to write one data item is 1/500mHz = 2ns
The total time required to write all data items in the burst is 450*2ns = 900ns
The time required to read one data item is 5/500ms = 10ns
In a time period of 900 ns, a total of 450 data items are written.
The number of data items that can be read within 900 ns is 900/10 = 90
The remaining number of data items that must be stored in the FIFO is 450-90 =360
Therefore, under this operating condition, the FIFO must be capable of storing at least 360 data items.
Calculated FIFO Depth: 512 entries (9-bit address)
Data Width: 8 bits per entry.

Verification Requirements

Primary Verification Level: RTL Module Level**

We are verifying the complete async_fifo module as an integrated system consisting of:

- Submodules: fifo_mem, write_ctrl, read_ctrl, ptr_sync (2 instances)
- Interface: write port, read port, reset signals (separate domains)
- Synchronization paths: R2W and W2R clock domain crossings

1.**Complete Functional Testing**: Module-level verification allows testing all features and interactions.

2. **Interface Completeness**: Both write and read domains fully exercised
3. **Integration Point**: Validates how submodules work together correctly
4. **Production Ready**: RTL verification ensures design correctness before synthesis
5. **CDC Validation**: Clock domain crossing can be verified end-to-end

Required Tools

Questa Sim: simulation tool.

High-Level Language : System Verilog

Risks and Dependencies

Risk 1: Metastability in CDC (Clock Domain Crossing)

- Threat: Synchronized pointers could violate metastability requirements
- Severity: High
- Mitigation:
    - Use proven two-stage flip-flop synchronizer architecture
    - Run extended simulations at clock edges to detect metastability
    - Insert intentional skew testing to verify robustness

Risk 2: Pointer Wraparound Edge Cases

- Threat: Gray code conversion errors at pointer boundaries
- Severity: Medium
- Mitigation:
    - Test full/empty conditions at ADDR_W bit boundaries (256, 384, 512 entries)
    - Verify Gray code logic independently before integration
    - Create targeted tests for wrap-around scenarios

Risk 3: Reset Sequence Timing

- Threat: Asynchronous reset signals may interact incorrectly with clock transitions
- Severity: Medium
- Mitigation:
    - Test reset during various clock phases in both domains
    - Verify flip-flops properly initialize to known states
    - Check flag outputs settle to expected values post-reset
    - Include reset-during-operation scenarios

Risk 4: Data Corruption from Simultaneous Read/Write

- Threat: Overlapping read and write operations could cause data loss
- Severity: High
- Mitigation:
    - Create extensive concurrent read/write test scenarios.
    - Monitor FIFO occupancy during mixed operations.
    - Use scoreboard to verify all data integrity.
    - Implement assertions for memory access conflicts.

Risk 5: Burst Size Limitations

- Threat: Cannot verify full 450-item burst on small ADDR_W values
- Severity: Low (mitigated by depth calculation)
- Mitigation:
    - Use calculated ADDR_W=9 (512 entries) to accommodate max burst

    - Create tests that exercise near-full and near-empty thresholds
    - Test multiple back-to-back bursts separated by idle cycles

Risk 6: Clock Frequency Variations

- Threat: Design assumes specific frequencies; variations could cause timing issues
- Severity: Low
- Mitigation:
    - Parameterize clock periods in testbench
    - Test at ±10% frequency variation
    - Verify synchronization latency remains acceptable
    - Stress test with worst-case frequency relationships

Dependencies

Internal Dependencies:

- Correct RTL implementation of all submodules
- Accurate pointer synchronization logic
- Proper Gray code generation and comparison
- Memory initialization and read/write data paths

Team Dependencies:

- Design team: Complete RTL implementation (Milestone 1)
- Verification team: Testbench development concurrent with RTL
- Integration: Verification environment ready by Milestone 2

Functions to be Verified.

Functions from specification and implementation

Function	Description
Write Operation	Data written to FIFO at write pointer location on wclk rising edge when wen=1 and wfull=0
Read Operation	Data read from FIFO at read pointer location asynchronously when ren=1 and rempty=0
Full Flag Generation	wfull asserted when write pointer (MSB inverted) matches synchronized read pointer
Empty Flag Generation	rempty asserted when read pointer matches synchronized write pointer

3/4 Full Threshold	Flag asserted when occupancy > 384 entries (75% of 512)
1/4 Empty Threshold	Flag asserted when occupancy < 128 entries (25% of 512)

Function	Description	Verification Method
R2W Synchronization	Read pointer synchronized from read domain to write domain via ptr_sync	Verify pointer stability after 2 wclk cycles
W2R Synchronization	Write pointer synchronized from write domain to read domain via ptr_sync	Verify pointer stability after 2 rclk cycles
Gray Code Encoding	Pointer values correctly converted to Gray code in each domain	Mathematical verification of encoding/decoding
Gray Code Comparison	Full/empty detected by comparing Gray code pointers correctly	Ensure MSB inversion logic for full detection

Resources requirements:

Our team's Final Project this term entails designing, implementing, and rigorously verifying an Asynchronous FIFO based on the provided High-Level Specification. Given the criticality of verification in ensuring design correctness, we have allocated significant time and resources towards verification, particularly UVM-based verification, within the tight timeframe of 6 weeks.

Schedule

For this project, all three of us are involved in both design and verification, but each person has a primary focus so that the work is well organized.

- Sai Charan will focus on the front-end of the project. He is responsible for writing the high-level design specification, choosing and justifying the FIFO depth for Option A, and

clearly defining the interfaces and flag behavior. Sai Charan will also help review the RTL and the verification environment when design questions come up.

- Steeva will focus mainly on the RTL. He will implement the async FIFO (memory, pointer logic, synchronizers), keep the design parameterized, and make sure it compiles and runs in Questa. Steeva will also bring up the initial "simple" testbench with clocks, resets, and a few basic write/read tests to sanity-check the design.
- Akshay will do verification. He will write and maintain the verification plan, build the class-based testbench (transactions, drivers, monitors, scoreboard), and later extend it into a UVM environment. Akshay will drive directed and constrained-random testing, add coverage and assertions, and track which scenarios have been verified.

Across the 6-week schedule, all three team members will collaborate on debugging issues, updating the documents, and preparing the final report and presentation.

Schedule and milestones

Milestone	Target Dates	Primary Owner	Key Deliverables	Success Criteria
M1: Design & Basic Verification	Week 1	SAI CHARAN, STEEVA	Design spec, RTL impl., Conv. TB, Depth calc.	RTL compiles, TB runs, basic write/read works
M1 Completion Check	End Week 1	AKSHAY	Verification plan v1	Verification strategy defined
M2: Class-Based Testbench	Week 2-3	AKSHAY , STEEVA	Transaction classes, drivers, monitors, 20-50 bursts	50+ random burst tests pass, >50% coverage
M2 Completion Check	End Week 3	AKSHAY	Updated verification plan	Coverage metrics documented
M3: Complete Coverage	Week 3-4	AKSHAY , STEEVA	Final class TB, coverage reports, RTL fixes	100% line coverage, 95%+ functional coverage
M3 Completion Check	End Week 4	SAI CHARAN	Design doc final review	All critical functions verified
M4: UVM Architecture	Week 4-5	AKSHAY	UVM testbench skeleton, agents, sequencer	UVM components compile, basic sequences work
M4 Completion Check	Mid Week 5	AKSHAY	UVM verification plan section	UVM architecture documented
M5: Final Verification & Delivery	Week 5-6	AKSHAY , STEEVA	Complete UVM TB, bug injection tests, final coverage	100% coverage, all scenarios pass,

				documentation complete
Final Delivery	End Week 6	Team	Zipped deliverables, presentations	All items submitted per project requirements

Github Repository Link : saibalne/ECE-593-ASYNC-FIFO-DV: Asynchronous FIFO Design and Verification

Resource

- Cliff Cummings. "Asynchronous & Synchronous Reset Design Techniques - Part Deux." SNUG San Jose, 2003.
- Cliff Cummings. "Simulation and Synthesis Techniques for Asynchronous FIFO Design." SNUG San Jose, 2002. Sunburst Design. "Asynchronous FIFO Design." Technical Documentation.
- .Accellera SystemVerilog Language Reference Manual (IEEE 1800-2017).
- Asynchronous FIFO Design – rfwireless-world.
- FIFO Architecture, Functions, and Applications – Texas Instruments.
- Asynchronous FIFO – VLSI Verify.
- Perplexity.ai