VERIFICATION TEST PLAN

Fundamentals of Pre-Silicon Validation
Winter -2024

Asynchronous FIFO

Ashwin Sivakumar, Mahidhar Regalla, Monica
Pinnamaneni

Date:02-02-2024

1 Table of Contents

2 Introduction:

2.1 Objective of the verification plan

The objective of this verification plan is to ensure the functional correctness and performance compliance of the FIFO design. Specifically, the plan aims to verify a FIFO with a depth of 140, read idle cycles of 3, and write idle cycles of 0.

2.2 Top Level block diagram
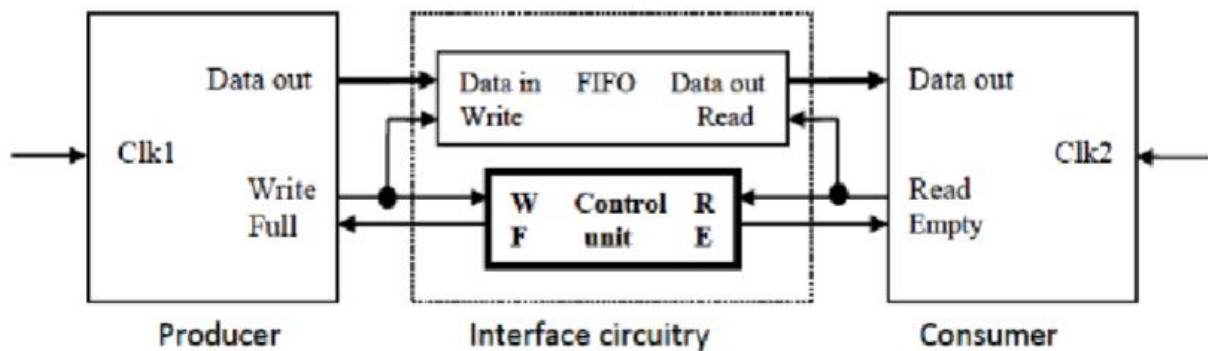
2.3 Specifications for the design
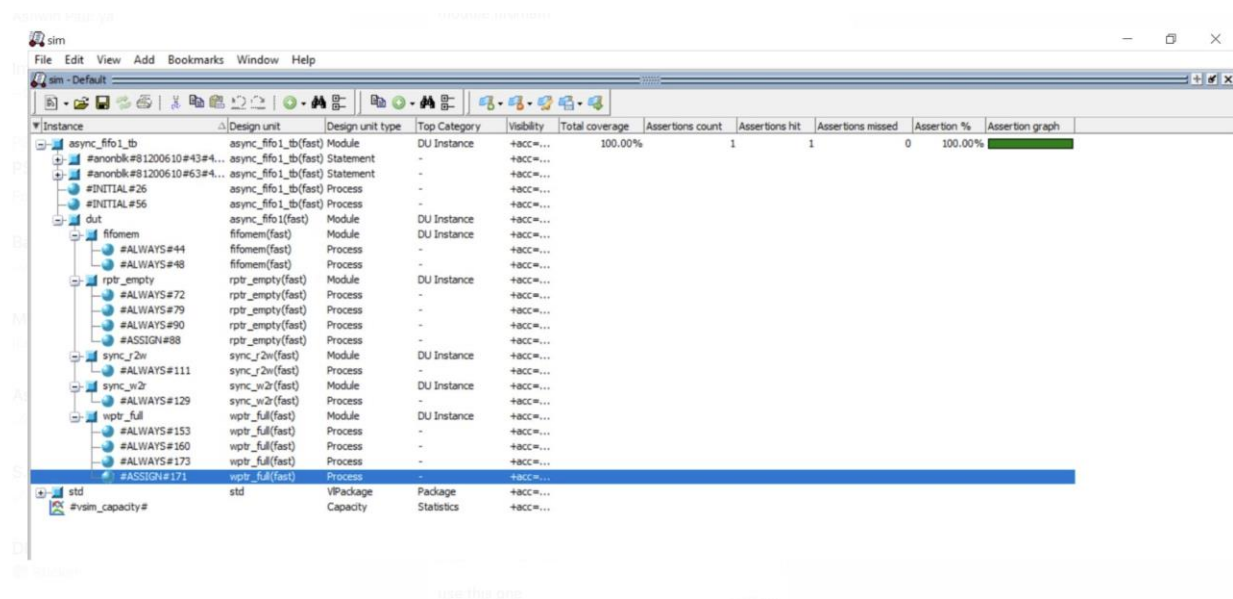
- Producer Freq = 250Mhz
- Consumer Freq = 100Mhz
- Write Idle Cycle= 0
- Read Idle Cycle= 3
- Burst Length = 150
- Duty-cycle= 50:50
- The FIFO specifications include a depth of 140, read idle cycles of 3, and write idle cycles of 0.

3 Verification Requirements

3.1 Verification Levels

3.1.1   Module hierarchy

We are verifying the FIFO design at the block level as it encapsulates the complete functionality of the FIFO.

3.1.2   Controllability and Observability

Controllability and observability are achieved through dedicated input and output ports of the FIFO module, allowing effective stimulus application and result monitoring.

3.1.3   Interface signals

FIFO Module Interface:

Definition: The primary interface of the FIFO module, including input and output ports.

Specifications:

winc: Write enable signal.

wclk: Write clock signal.

wrst_n: Active-low write reset signal.

rinc: Read enable signal.

rclk: Read clock signal.

rrst_n: Active-low read reset signal.

wdata: Input data bus for write operations.

rdata: Output data bus for read operations.

wfull: Output signal indicating FIFO full condition.

rempty: Output signal indicating FIFO empty condition.

Memory Interface (fifomem):

Definition: The interface between the FIFO module and the memory subsystem (fifomem).

Specifications:

winc: Write enable signal.

wfull: Input signal indicating FIFO full condition.

wclk: Write clock signal.

waddr: Write address bus.

raddr: Read address bus.

wdata: Input data bus for write operations.

rdata: Output data bus for read operations.

Read Pointer Empty Module Interface (rptr_empty):

Definition: The interface of the module handling read pointer and empty condition.

Specifications:

rinc: Read enable signal.

rclk: Read clock signal.

rrst_n: Active-low read reset signal.

rq2_wptr: Input read pointer with write side.

rempty: Output signal indicating FIFO empty condition.

raddr: Output read address.

Write Pointer Full Module Interface (wptr_full):

Definition: The interface of the module handling write pointer and full condition.

Specifications:

winc: Write enable signal.

wfull: Output signal indicating FIFO full condition.

wclk: Write clock signal.

wq2_rptr: Input read pointer with write side.

waddr: Output write address.

wptr: Output write pointer.

Synchronization Modules (sync_r2w, sync_w2r):

Definition: Modules ensuring synchronization between read and write pointers.

Specifications:

wclk, wrst_n: Write clock and reset signals.

rptr: Read pointer.

rq2_wptr, wq2_rptr: Synchronized read and write pointers.

4 Required Tools

4.1 Software and hardware tools.

Questasim.

4.2 Directory structure of your runs, what computer resources you will be using.

The directory structure for the verification runs will be organized for clarity and ease of access. Below is a proposed directory structure:

verification_plan

  /testbench

    - async_fifo1_tb.sv

/modules

    - async_fifo1.sv

    - fifomem.sv

    - rptr_empty.sv

    - wptr_full.sv

    - sync_r2w.sv

    - sync_w2r.sv

5 Risks and Dependencies

Incomplete Specifications:

Risk: Incomplete or ambiguous specifications may lead to misinterpretations during verification.

Contingency/Mitigation: Regularly communicate with design and specification teams to clarify any ambiguities. Document assumptions made during verification.

Concurrency Issues:

Risk: Concurrency-related problems may arise during read and write operations.

Contingency/Mitigation: Thoroughly test scenarios involving concurrent read and write operations to identify and address any concurrency issues.

Performance Bottlenecks:

Risk: Performance issues, such as slow read or write operations, may impact the overall functionality.

Contingency/Mitigation: Implement performance-related tests to ensure compliance with specifications. Optimize design if performance bottlenecks are identified.

Integration Challenges:

Risk: Integration issues may arise when connecting the FIFO design with other system components.

Contingency/Mitigation: Conduct integration testing with other modules early in the verification process. Collaborate with other teams to address interface compatibility.

Assertion Failures:

Risk: Assertions may fail to capture specific corner cases or may trigger false positives.

Contingency/Mitigation: Regularly review and update assertions. Use multiple assertion types to cover various aspects of the design.

Incomplete Test Coverage:

Risk: Insufficient test coverage may lead to undetected bugs or corner cases.

Contingency/Mitigation: Develop a comprehensive test plan with a variety of scenarios to achieve high test coverage. Utilize code coverage tools to identify areas that need additional testing.

Misalignment with Design Changes:

Risk: Changes in the design may not be accurately reflected in the testbench or test cases.

Contingency/Mitigation: Establish a robust change management process. Regularly synchronize the testbench and test cases with the latest design specifications.

Resource Constraints:

Risk: Insufficient computing resources may impact the efficiency of simulations.

Contingency/Mitigation: Optimize testbenches for efficiency. Use parallel simulation options if available. Upgrade hardware resources if necessary.

Model-Reality Mismatch:

Risk: The model may deviate from the real-world behavior, leading to incorrect verification results.

Contingency/Mitigation: Regularly validate the model against real-world scenarios. Update the model based on feedback from silicon testing.

Incomplete Documentation:

Risk: Lack of comprehensive documentation may hinder understanding and troubleshooting.

Contingency/Mitigation: Maintain up-to-date documentation. Document assumptions, known issues, and resolutions for future reference.

Ambiguous Error Handling:

Risk: Unclear error handling mechanisms may make it challenging to identify and debug issues.

Contingency/Mitigation: Clearly define error handling procedures. Implement detailed logging and reporting mechanisms.

Inadequate Regression Testing:

Risk: Changes to the design or testbench may introduce regressions that go undetected.

Contingency/Mitigation: Implement a comprehensive regression testing suite. Verify the integrity of existing functionalities after each update.

Unrealistic Test Scenarios:

Risk: Test scenarios may not accurately represent real-world usage.

Contingency/Mitigation: Collaborate with domain experts to create realistic test scenarios. Incorporate feedback from system architects.

Human Error:

Risk: Mistakes in testbench development, test case creation, or result analysis.

Contingency/Mitigation: Implement thorough code reviews, use automation where possible, and encourage a culture of careful verification practices.

6 Functions to be Verified.

6.1 Functions from specification and implementation

6.1.1 List of functions that to be verified.

Write Operation:

Function: Verify that data can be successfully written into the FIFO.

Description: Ensure that the FIFO accepts data when the write enable signal is asserted. Check if the data is correctly stored in the memory.

Read Operation:

Function: Verify that data can be successfully read from the FIFO.

Description: Ensure that the FIFO provides valid data when the read enable signal is asserted. Check if the data read matches the expected values.

Write and Read Concurrency:

Function: Verify simultaneous write and read operations.

Description: Test scenarios where write and read operations occur concurrently. Ensure that the FIFO handles simultaneous read and write requests without data corruption.

FIFO Full Condition:

Function: Verify the FIFO full condition.

Description: Write data into the FIFO until it reaches its maximum depth. Verify that the FIFO signals a full condition correctly.

FIFO Empty Condition:

Function: Verify the FIFO empty condition.

Description: Read data from the FIFO until it becomes empty. Verify that the FIFO signals an empty condition correctly.

Idle Write Cycles:

Function: Verify the behavior during idle write cycles.

Description: Confirm that the FIFO remains stable during idle write cycles (when no data is being written). Check for any unintended side effects during idle write periods.

Idle Read Cycles:

Function: Verify the behavior during idle read cycles.

Description: Confirm that the FIFO remains stable during idle read cycles (when no data is being read). Check for any unintended side effects during idle read periods.

Write Error Handling:

Function: Verify error handling during write operations.

Description: Test scenarios where the FIFO receives write requests beyond its capacity. Ensure that appropriate error handling mechanisms are in place.

Read Error Handling:

Function: Verify error handling during read operations.

Description: Test scenarios where read requests are made when the FIFO is empty. Ensure that appropriate error handling mechanisms are in place.

Reset Operation:

Function: Verify the reset functionality.

Description: Assert the reset signal and verify that the FIFO resets to its initial state, clearing all stored data.

Sequential Write and Read:

Function: Verify sequential write and read operations.

Description: Perform a series of sequential write and read operations to ensure proper data flow through the FIFO.

Asynchronous Write and Read:

Function: Verify asynchronous write and read operations.

Description: Introduce variations in write and read timing to ensure that the FIFO can handle asynchronous operations.

6.1.2   List of functions that will not be verified.

Power-On Self-Test (POST):

Description: The power-on self-test is not verified.

Reason: POST is typically a one-time initialization process that occurs during power-up. Since it is not part of the regular operation of the FIFO, it is not explicitly verified.

External Clock Source Handling:

Description: Verification of handling external clock sources is not performed.

Reason: The handling of external clock sources is often assumed to be functioning correctly at the system level. Detailed verification at the FIFO level may not be necessary.

Integration with Higher-Level Modules:

Description: Interaction and integration with higher-level modules are not explicitly verified.

Reason: This verification plan focuses on the functional aspects of the FIFO design. Integration with other modules is often verified at a higher level of the system.

Environmental Conditions (Temperature, Voltage, etc.):

Description: The FIFO's behavior under extreme environmental conditions is not verified.

Reason: Environmental conditions are usually part of the system-level testing. Detailed verification of extreme conditions may be addressed at the system level rather than at the FIFO level.

Electromagnetic Interference (EMI) Compliance:

Description: Verification of EMI compliance is not performed.

Reason: EMI compliance is typically a system-level consideration, and specific verification is often carried out during system integration.

Security Features:

Description: Verification of security features is not explicitly performed.

Reason: Security features, if present, are often considered at the system or chip level. This plan focuses on functional verification rather than security aspects.

User Interface Aspects:

Description: Verification of user interface aspects such as graphical interfaces is not performed.

Reason: User interfaces are often part of the higher-level system, and their verification is typically carried out at that level.

FIFO Physical Design Aspects (Layout, Placement, Routing):

Description: Physical design aspects such as layout, placement, and routing are not verified.

Reason: These aspects are typically addressed during the physical design and layout stages rather than in the functional verification plan.

Regulatory Compliance (e.g., FCC Compliance):

Description: Verification of regulatory compliance is not explicitly performed.

Reason: Regulatory compliance is usually a system-level consideration, and verification may involve compliance testing after integration.

Manufacturing Test Coverage:

Description: Verification of specific manufacturing test coverage is not performed.

Reason: Manufacturing test coverage is often addressed separately in manufacturing test plans and may not be part of functional verification.

User Documentation and Manuals:

Description: Verification of user documentation and manuals is not explicitly performed.

Reason: User documentation is typically reviewed and validated separately from functional verification.

Obsolete or Deprecated Features:

Description: Verification of features that are obsolete or deprecated is not performed.

Reason: Deprecated features are often excluded from active development and are not part of the functional verification scope.

6.1.3 List of critical functions and non-critical functions for tapeout

List of Critical Functions for Tapeout:

Write Operation Integrity:

Scenario: Verify that the FIFO correctly writes data into the memory under normal operating conditions.

Expected Outcome: Successful write operations with data integrity maintained.

Read Operation Integrity:

Scenario: Ensure that the FIFO reads data accurately from memory without corruption.

Expected Outcome: Successful and accurate retrieval of data during read operations.

FIFO Full and Empty Conditions:

Scenario: Verify that the FIFO signals appropriately when it is full and empty.

Expected Outcome: Proper signaling of full and empty conditions without false indications.

Read and Write Synchronization:

Scenario: Confirm that read and write operations are synchronized and occur without data hazards.

Expected Outcome: No data hazards, ensuring synchronized read and write operations.

Idle Cycles Handling:

Scenario: Validate the behavior of the FIFO during idle cycles, especially considering read and write idle cycles.

Expected Outcome: Correct handling of idle cycles as per specifications.

Addressing and Pointers:

Scenario: Ensure proper addressing and handling of pointers within the FIFO memory.

Expected Outcome: Accurate addressing and pointer manipulation without errors.

Reset Operation:

Scenario: Verify the functionality of the reset operation, including its impact on pointers and memory.

Expected Outcome: Successful reset with proper initialization of internal states.

Concurrent Read and Write:

Scenario: Simulate concurrent read and write operations to check for any conflicts or race conditions.

Expected Outcome: Proper handling of concurrent read and write operations without data corruption.

Power Consumption and Efficiency:

Scenario: Assess the power consumption and efficiency of the FIFO under various operational conditions.

Expected Outcome: Power consumption within specified limits and efficient use of resources.

Error Detection and Correction:

Scenario: Inject errors into the data and verify the FIFO's ability to detect and correct errors.

Expected Outcome: Successful detection and correction of errors without compromising system integrity.

List of Non-Critical Functions for Tapeout:

Asynchronous Clock Handling:

Scenario: Simulate variations in the asynchronous clock and verify its impact.

Expected Outcome: Non-critical as the FIFO may not be designed to handle extreme asynchronous clock variations.

Extreme Environmental Conditions:

Scenario: Subject the FIFO to extreme temperature and voltage conditions.

Expected Outcome: Non-critical for tapeout as extreme conditions are often addressed at the system level.

Overclocking Scenarios:

Scenario: Test the FIFO under overclocking conditions beyond specified limits.

Expected Outcome: Non-critical as tapeout focuses on normal operational conditions.

Unused or Reserved Features:

Scenario: Verify features marked as unused or reserved in the specifications.

Expected Outcome: Non-critical, as these features may not impact normal functionality.

Backward Compatibility:

Scenario: Assess backward compatibility with previous versions of the FIFO.

Expected Outcome: Non-critical for tapeout unless backward compatibility is explicitly required.

Interoperability with Legacy Systems:

Scenario: Check interoperability with legacy systems or older versions of components.

Expected Outcome: Non-critical unless specified for the system's operational context.

User Interface Aspects:

Scenario: Evaluate graphical user interface aspects or other non-functional features.

Expected Outcome: Non-critical for tapeout as these aspects are typically validated separately.

Advanced Security Features:

Scenario: Verification of advanced security features beyond basic data integrity checks.

Expected Outcome: Non-critical unless advanced security features are a primary focus.

Obsolete or Deprecated Features:

Scenario: Verification of features marked as obsolete or deprecated.

Expected Outcome: Non-critical as these features are not actively developed.

Regulatory Compliance Testing:

Scenario: Conduct regulatory compliance testing beyond basic functional verification.

Expected Outcome: Non-critical for tapeout, as specific compliance testing may occur at later stages.

7 Tests and Methods

7.1.1 Testing methods to be used: Black/White/Gray Box.

Black Box Testing: Functional verification based on specifications.

White Box Testing: Code coverage analysis and assertion-based verification.

Gray Box Testing: Scenario-based testing for corner cases.

7.1.2 PROs and CONs.

Black Box Testing: PRO - High-level coverage. CON - Limited visibility into internals.

White Box Testing: PRO - In-depth analysis. CON - May miss system-level issues.

Gray Box Testing: PRO - Comprehensive testing. CON - Increased simulation time.

7.1.3 Testbench Architecture; Component used (list and describe Drivers, Monitors, scoreboards, checkers etc.)

www.vlsiverify.com
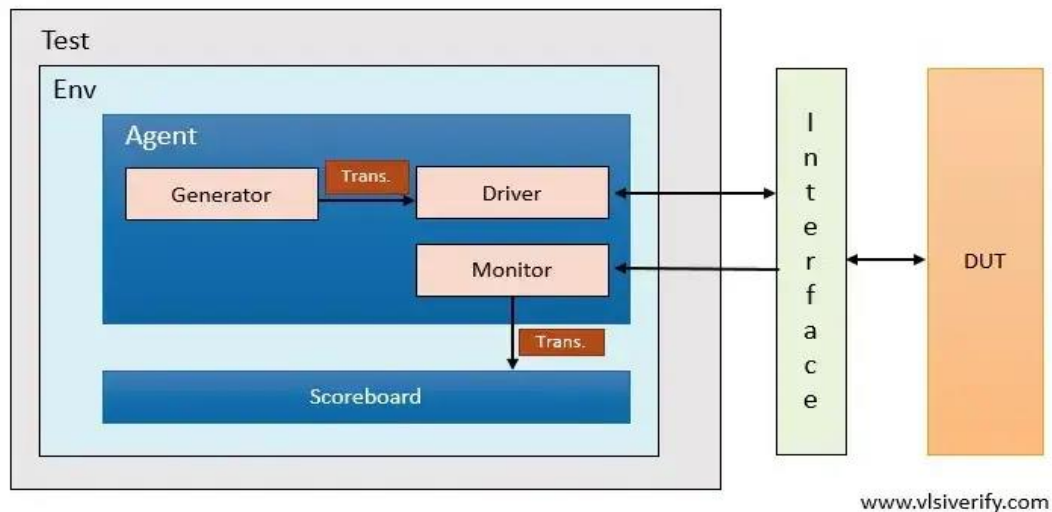
Testbench components

The testbench module **async_fifo1_tb** initializes and drives signals to the DUT and checks the output against expected results. To relate this code to the image, we could map the code components to the diagram as follows:

- **Generator**: In the testbench (**async_fifo1_tb**), the part where random data is generated (**$urandom**) and controlled by **winc** can be seen as the generator.

- **Driver**: The signals **winc**, **wdata**, **wclk**, and **wrst_n** driven from the testbench to the DUT represent the driver.

- **Monitor**: The part of the testbench that reads the output **rdata** and compares it with the expected data (**verif_wdata**) acts as the monitor.

- **Scoreboard**: The array **verif_data_q** and the assertions checking the DUT's output (**rdata**) against the expected output (**verif_wdata**) can be considered as the scoreboard mechanism.

- **Interface**: This is implicit in the testbench, as the signals driven to the DUT and the signals read from it act as the interface.

7.1.4 Verification Strategy: (Dynamic Simulation, Formal Simulation, Emulation etc.) Describe why you chose the strategy.

Verification Strategy: Dynamic Simulation.

Reasoning: Dynamic simulation balances accuracy and efficiency for FIFO verification.

7.1.5 Driving Methodology

7.1.5.1 List the test generation methods (Directed test, constrained random)
Test Generation Methods: DirectedTesting and Constraint random testing

Direct Testing involves creating targeted test cases based on specific design requirements, while Constraint Random Testing (CRT) uses randomized stimulus generation guided by constraints to explore the design space more thoroughly. Both techniques play vital roles in verifying digital designs, with Direct Testing focusing on known requirements and CRT efficiently exploring diverse test scenarios to uncover corner cases and unexpected behaviors

7.1.6 checking methodology

1. **Self-Checking Testbench**: Automates the process of stimulus generation and response verification without manual intervention, using a testbench that compares the DUT's outputs against expected results.

2. **Assertion-Based Checking**: Employs conditional checks within the simulation to ensure the DUT's behavior aligns with specified design properties and constraints.

3. **Scoreboarding**: Utilizes a reference model to predict outputs, facilitating comparison against actual DUT responses for validation of correct behavior.

4. **Coverage Analysis**: Measures the extent to which the DUT's state space is exercised by the testbench, aiming to identify untested parts of the design.

5. **Randomized Testing**: Generates random inputs to stress-test the DUT across a broad range of scenarios, often uncovering edge cases not considered in directed testing.

6. **Directed Testing**: Focuses on specific and critical DUT functionalities by crafting targeted test cases, ensuring that particular behaviors are correctly implemented.

7. **Simulation Control**: Manages the execution flow of the simulation environment, such as clock cycles and reset sequences, to emulate realistic operating conditions.

8. **Logging and Error Reporting**: Captures and outputs simulation data and error messages to track test progress and facilitate debugging of failed checks.

7.1.6.1 From specification, from implementation, from context, from architecture etc.

From Specification.

7.1.7 Testcase Scenarios (Matrix)

7.1.7.1 Basic Tests

Test Name / Number	Test Description/ Features
1.1.1	Check basic read operation
1.1.2	Check basic write operation
1.1.3	Check FIFO Full condition
1.1.4	Check FIFO empty condition
1.1.5	Reset Operation

7.1.7.2   Complex Tests

Test Name / Number	Test Description/ Features
1.2.1	Concurrent events (R+W) Conditions: fifo_full/fifo_empty/always_full/always empty etc.
1.2.2	Idle Cycles Handling Validate Idle cycles between read and write operations.
1.2.3	Error detection and Correction Inject errors and check behavior of FIFO.

7.1.7.3   Regression Tests (Must pass every time)

Test Name / Number	Test Description/Features
1.3.1	Basic read operation
1.3.2	Basic write operation
1.3.3	Reset Operation
1.3.4	FIFO full condition
1.3.5	FIFO empty condition

7.1.7.4   Any special or corner cases testcases

Test Name / Number	Test Description
1.4.1	Minimum FIFO Depth
1.4.2	Bug injection and testing scenario

8   Coverage Requirements

8.1.1.1   Describe Code and Functional Coverage goals for the DUV.

Address Coverage:

Goal: Achieve 100% coverage of all addressable memory locations.

Conditions:

Ensure that all memory locations in the FIFO are accessed during the simulation.

Verify that there are no untested or unreachable areas of memory.

Data Width Coverage:

Goal: Achieve coverage for all possible data widths.

Conditions:

Perform read and write operations with various data widths.

Ensure that each supported data width is adequately tested.

Boundary Value Coverage:

Goal: Achieve coverage at boundary values for address and data width.

Conditions:

Test read and write operations at the maximum and minimum values for address and data width.

FIFO Full and Empty Conditions:

Goal: Achieve coverage for FIFO full and empty conditions.

Conditions:

Simulate scenarios where the FIFO is full or empty and ensure proper handling.

Read and Write Idle Cycles:

Goal: Achieve coverage for read and write idle cycles.

Conditions:

Introduce idle cycles during read and write operations and confirm correct handling.

8.1.1.2   Conditions to achieve coverage goals.

Covergroups and Coverpoints:

Explanation: Define covergroups and coverpoints to capture critical aspects of the FIFO operation.

Conditions:

Utilize covergroups to track address coverage, data width coverage, and FIFO status conditions.

Define coverpoints to capture specific scenarios such as full and empty conditions.

Selection of Bins:

Explanation: Specify bins within covergroups to categorize simulation outcomes.

Conditions:

Use bins to differentiate between various address values, data widths, and FIFO status conditions.

Ensure that each bin corresponds to a specific functional aspect of the FIFO.

8.1.2 Assertions

8.1.2.1 Describe the assertions that you are planning to use and how it will help you improve the overall coverage and functional aspects of the design.

FIFO Full and Empty Assertions:

Description: Assertions to check FIFO full and empty conditions.

Conditions:

Implement assertions to monitor and verify that the FIFO behaves correctly when full or empty.

Data Integrity Assertions:

Description: Assertions to ensure data integrity during read and write operations.

Conditions:

Use assertions to compare expected and observed data values, checking for consistency.

Address Range Assertions:

Description: Assertions to verify that memory accesses stay within the valid address range.

Conditions:

Implement assertions to check that all memory accesses fall within the expected address range.

Idle Cycle Assertions:

Description: Assertions to confirm correct handling of read and write idle cycles.

Conditions:

Use assertions to validate that the FIFO behaves appropriately during idle cycles.

9 Resources requirements

9.1 Team members and who is doing what and expertise.

No	Task	Actual Start Date	Responsible person	Target Completion date & Status		Comments
1	High Level Design Specification (HLDS)	19/1/2024	Ashwin, Mahidhar & Monica	22/1/24	completed	Design spec documentation
1.1	Design spec calculation and plan	21/1/2024	Ashwin, Mahidhar & Monica	23/1/4	Completed	Depth calculation modules understanding, and plan has been made for the design and modules has been splitted.
1.2	The sunburst paper review	23/1/4	Ashwin, Mahidhar & Monica	23/1/24	Completed	The sunburst paper has been reviewed and design implementation method has been decided accordingly
2	Design Implementation	23/1/24	Ashwin, Mahidhar & Monica	8/2/24	progress	
2.1	Fifo mem module	24/1/24	Ashwin	25/1/24	completed	System verilog code has been implemented for FIFO mem module
2.2	Fifo Write to Read pointer	26/1/24	Ashwin	27/1/24	completed	System verilog code has been implemented and updated in github
2.3	FIFO full	27/1/24	Monica	29/1/24	completed	Sv code has been implemented.

2.4	Fifo read to write pointer	24/1/24	Mahidhar	25/1/24	completed	Sv code has been implemented.
2.5	FIfo empty	26/1/24	Mahidhar	27/1/24	completed	Sv code has been implemented.
2.6	Fifo Top	27/1/24	Monica	30/1/24	completed	Sv code has been implemented.

2.7	Fifomem testing	29/1/24	Ashwin	30/1/24	completed	The testbench with assertions has been created for fifomem has been tested.
2.8	Fifomem testbench plan	30/1/24	Ashwin	31/1/24	completed	The testbench plan has been written and testcases

10 Schedule

- Weeks 1-2 (Current Date to January 20, 2024): Focus on developing the HLDS. This includes finalizing specifications and planning out the design and verification strategy.

- Weeks 3-5 (January 21 to February 10, 2024): Begin the implementation of the Async FIFO in SystemVerilog. Ensure that the design meets the requirements set out in the HLDS.

- Weeks 6-7 (February 11 to February 20, 2024): Develop the initial conventional testbench and begin basic testing of the design.

- Weeks 8-9 (February 21 to February 28, 2024): Expand into functional verification using UVM. This includes developing a more sophisticated UVM testbench and achieving the desired coverage goals.

- Week 10 (March 1 to March 5, 2024): Perform final testing and validation of the design. Address any bugs or issues found during testing.

- Week 11 (March 6 to March 7, 2024): Finalize documentation, prepare all deliverables, and package everything for submission.

11 References Uses / Citations/Acknowledgements

1.http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO1.pdf
2.http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO2.pdf
3.https://hardwaregeeksblog.files.wordpress.com/2016/12/fifodepthcalculationmadeeasy2.pdf
4.https://ieeexplore.ieee.org/abstract/document/7237325

5.https://d1wqtxts1xzle7.cloudfront.net/56108360/EC109-libre.pdf?1521537519=&response-content-disposition=inline%3B+filename%3DDesigning_Asynchronous_FIFO.pdf&Expires=1705290689&Signature=QzZekKdAbEuISN-ntg8apOHJqOd6HgWlX7JNpkhTDLi85ymDBCRoPZUAFVVzLieCjYWWnHMnguKyUKfXV9wosjPxiixlajsjvW0Y-IaVol9q9zXSg~KcmI9UWZLFzahEnUb2ZgRrEWeNEu2ZKotlW8UFJbpZwVpcdQgRVAzeEu1fZ1hgxo9n42-jZrkO8e3BroHDbs8k5~6Vgpq9~B45KsInZuOS469bksmH22x9pIOUJCXHVnyoSGqKW7AaxA88jwabqRfuK2nDPmTwewufpkUWiUqm-~7klUrS4YjlAS7rc9kWw82Q3KPpNAg7VBxf9J05h7k2FqmNHQvomW4BDA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

6.https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6041338

7.https://www.youtube.com/watch?v=UNoCDY3pFh0

8.Chat gpt