ECE593
Verification Plan

Sanket Patil (sanket2@pdx.edu) & Balaji Rao Vavintaparthi (balaji2@pdx.edu)

Verification Requirements

A. Verification Levels:

Unit Level

B. Functions:

• Reset	FIFO will be empty
• Write	FIFO will get new entry
• Read	FIFO will lose the first written entry
• Clear	FIFO will lose the last written entry
• Bypass	FIFO will remain same
• Error	FIFO will set the error signal high for error scenarios

C. Specific Tests and Methods:

Type of Verification:

It's black box testing as we are checking through reference models which monitors only the input and output signals.

It’s also a white box testing as we have used assertions to verify the design in which we also have accessed our internal signals.

Verification Strategy:

Verified the design by creating a class-based testbench environment. We have a random_stimulus class in which we are randomizing the write data. We have tester class in which we are calling the tasks randomly. But the tasks are directed except write data. We also have a checker class for verifying the design which includes assertion and reference model.

Abstraction level:

The verification strategy implements a Behavioural Level Abstraction

Checking:

Assertions:

Reset without error: When RESET signal is asserted and all remaining signals are deasserted, all the signals should transition to 0 in the subsequent cycle.

Reset with error: In the event that both the RESET signal and another signal are asserted simultaneously, the error signal should transition to a high state within the same clock cycle.
Write when FIFO is full: When Write is asserted write pointer shouldn't change when FIFO is full
Write when FIFO is not full: When write is asserted write pointer should increment when FIFO is not full
Error for Write when FIFO is full: When Write is asserted FIFO should be full when pointers are same and write is asserted
Read when FIFO is empty: When Read is asserted read pointer should not change when FIFO is empty
Read when FIFO is not empty: When Read is asserted read pointer should increment when FIFO is not empty
Error for read when FIFO is empty: When Read is asserted Error signal should be asserted when fifo is empty
Read when FIFO is full: When Read is asserted FIFO should not be full
Read for read data: When Read is asserted Read data should be known when FIFO is not empty
Read pointer and empty: When Read is asserted FIFO should be empty when pointers are same position
Clear for FIFO empty: When Clear is asserted write pointer should not change when FIFO is empty
Clear when FIFO isn't empty: When Write is asserted write pointer should increment when FIFO is not full
Error for Clear when FIFO empty: When Clear is asserted error signal should be asserted when FIFO is empty
Clear for FIFO full: When Clear is asserted FIFO should not be full
Clear pointer empty: When Read is asserted FIFO should be empty when pointers are same
Bypass: Read data should be equal to write data
Bypass pointer: Pointers do not change

D. Coverages:

We are doing functional coverage.
Cover points for

Reset	where Reset=0 and Reset=1
rd_en	where rd_en =0 and rd_en =1
wr_en	where wr_en =0 and wr_en =1
clear	where clear =0 and clear=1
fifo_empty	where fifo_empty =0 and fifo_empty =1

fifo_full	where fifo_full=0 and fifo_full=1
rd_data	where rd_data=0, rd_data= ‘1, rd_data=default
wr_data	where wr_data=0, wr_data= ‘1, wr_data=default
error	where error=0 and error=1

Also the cross coverage of all the above cover points.

For back-to-back cases cross between all control signals and also with FIFO full, empty and error signals.

E. Scenario:

Reset: Checked the RESET by making reset signal high and the output should be ‘xx’, fifo should be empty and both pointers should point at zero.

Write: Checked the WRITE operation by making write enable high and read enable low and the fifo empty signal should be low, write pointer should increment.

Read: Checked the READ operation by making read enable high and write enable low and output should be the data pointed by read pointer, fifo high signal should be low and read pointer will be increment.

Clear: Checked the CLEAR operation by making clear signal high and write enable and read enable low and the fifo empty signal should be low, write pointer should decrement.

Bypass: Checked the BYPASS condition by making write enable and read enable high when FIFO is empty. The fifo empty signal should be asserted, pointers should remain same and output should be the write data.

Reset & other control signals high: Checked this condition by making Reset high along with other control signals high in which the error signal should get asserted.

FIFO Full & Write: Checked this condition by making write enable high when FIFO is full. Full fifo signal should be asserted and pointers should not change and error signal should be high.

FIFO Empty & Read: Checked this condition by making read enable high when FIFO is empty. Empty FIFO signal should asserted and pointers should not change and error signal should be high.

FIFO Empty & Clear: Checked this condition by making clear signal high when FIFO is empty. Empty FIFO signal should asserted and pointers should not change and error signal should be high.

Write & Clear: Checked this condition by making write enable and clear signal high and nothing will be changed and error signal should be high.

Read & Clear: Checked this condition by making read enable and clear signal high and nothing will be changed.

Write & Read & Clear: Checked this condition by making write and read enable and clear signal high and nothing will be changed.

Back-to-back Read: Multiple read operations back-to-back.

Back-to-back Write: Multiple write operations back-to-back.

Back-to-back Clear: Multiple clear operations back-to-back.
Back-to-back Write & Read: Checked by making write enable high and then read enable high after write operation is done. FIFO will get data in write operation and it will lose its entry in read operation.
Back-to-back Write & Clear: Checked by making write enable high and then clear high after write operation is done. FIFO will get data in write operation and it will lose its entry in clear operation.
Back-to-back Write & Reset: Checked by making write enable high and then reset high after write operation is done. FIFO will get data in write operation and it will do reset operation.
Back-to-back Read & Write: Checked by making read enable high and then write enable high after read operation is done. It will read the existing FIFO entried and then write in the FIFO.
Back-to-back Read & Clear: Checked by making read enable high and then clear high after read operation is done. It will read the existing FIFO entried and then clear one entry in the FIFO.
Back-to-back Read & Reset: Checked by making read enable high and then reset high after read operation is done. It will read the existing FIFO entried and then reset the FIFO.
Back to back Clear & Write: Checked by making clear high and then write enable high after clear operation is done. It will clear the existing FIFO entried and then write in the FIFO.
Back to back Clear & Read: Checked by making clear high and then read enable high after clear operation is done. It will clear the existing FIFO entried and then read the data in the FIFO.
Back-to-back Clear & Reset: Checked by making clear high and then read enable high after clear operation is done. It will clear the existing FIFO entried and then read the data in the FIFO.
Back-to-back Reset & Write: First did the reset operation and after that write enable was made high to write the entry in the FIFO.
Back-to-back Reset & Read: First did the reset operation and after that read enable was made high to read the entry in the FIFO.
Back-to-back Reset & Clear: First did the reset operation and after that clear was made high to clear the entry in the FIFO.

Reference Model:

We have a checkers class in which we have a task that will execute our reference model. In that class, we have two forever blocks that run simultaneously as we have used fork-join. The first forever block is executed at every negedge of the clock and the second is executed at the posedge of the clock. In the first forever block we are focusing on READ and BYPASS conditions as they both happen in the same cycle. Also, the proper error values are set for every condition. Fifo empty signal is also monitored in this forever block. In the second forever block we are focusing on WRITE, CLEAR, and RESET conditions. Fifo full is monitored in this block. After the 1st read which happens at negedge, the following reads are monitored in this block.

The need of two forever block is to obtain the cycle accuracy. We found few bugs in the RTL and we also fixed it. As of now we are running 1000 random sequence and every assertion is passing as well as reference model is also working fine. So, using reference model and assertions the bug founds are removed.

Project Management

A. Tools:

1. QuestaSim for sample coding, simulation and verification
2. MobaXTerm for server and SSH access
3. EDA playground
4. Notepad++

B. Risks/Dependencies:

The verification strategy that we used here employs Random Test Case Stimulus, which carries the risk of leaving some test scenarios undiscovered. Moreover, the design should be functional for various FIFO sizes.

C. Resources:

1. Project Description
2. Comprehensive Functional Verification: The complete industry cycle by Bruce Wile, John C. Gross, Wolfgang Roesner.
3. Course Slides provided by Prof. Alex J Olson
4. IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language, which includes a section on FIFOs: https://ieeexplore.ieee.org/document/7734042

D. Schedule:

WEEK	**PLAN**
1-2	Project 1 - FIFO Design
3-4	Project 2 - Bypass logic and Coverages
5-6	Project 3 - Error logic and Class-based TB
7-8	Project 4 - Checker Module and Assertions