VERIFICATION TEST PLAN
ECE 593 – P4

Abhigna Bheemineni
Manikandeshwar Sasidhar

1. VERIFICATION REQUIREMENTS

A. VERIFICATION LEVELS:

The design consists of a FIFO array, combinational logic for the empty, full, and output data signals, and sequential logic for updating the read and write pointers. Verification can however take place on a higher (instruction) level, at the top module.

B. FUNCTIONS:

SIGNAL(S)	COMMAND	FUNCTION
RESET	Reset Design	Clear the entire FIFO by setting the elements to ‘bx. Set the read and write pointers to zero and the FIFO_empty to one.
wr_en	Write	Write the wr_data into the FIFO at the location pointed by the write pointer provided that the FIFO is not full.
rd_en	Read	Read an element in the FIFO, who’s location is pointed by the read pointer, provided that the FIFO is not empty.
clear	Undo	Delete the most recently written element from the FIFO provided that the FIFO is not empty.
wr_en && rd_en	Bypass	Output the current input data if the FIFO was empty before this operation.

C. SPECIFIC TESTS & METHODS:

i. **Type of Verification:**

We have primarily used Black box checking (Reference model) to verify the DUT but we have provided supplementary white box checking via assertions.

ii. Verification Strategy:

Simple Directed and Random Tests: Tests to verify the basic working of the FIFO, pointers, full & empty signals, and output data.

Advanced Directed and Random Tests: Testing the commands with respect to other commands and targeting the numerous corner case instances.
Constraint Random Testing: Both the data and commands are randomized, and the design is tested, running against our golden reference (yet).

Assertions: Concurrent assertions to verify the inner workings of the design.

iii. **Abstraction Level:** Packet Level

iv. **Checking:**

WHITE BOX CHECKING:

Concurrent assertions were written to catch any anomalies in the design.

Sr. No.	**Assertions**	**Comments**
1.	When FIFO is empty and wr_en is HIGH then fifo_empty goes from HIGH to LOW	8 errors in random testing
2.	When FIFO is full and rd_en is HIGH then fifo_full goes from HIGH to LOW	PASSED
3.	When FIFO is full and clear is HIGH then fifo full goes from HIGH to LOW	PASSED
4.	When FIFO is empty then rdptr must not change	15 errors in random testing. Caused by doing illegal clear with reads
5.	When FIFO is full then wrptr must not change	2 errors in direct testing. Wrptr gets set to 0 if clear is asserted in some very niche cases when fifo is full
6/7	Fifo full and fifo empty cannot be asserted at the same time	PASSED
8.	Bypass logic checking	2 errors in direct testing. Multiple errors in random testing. Bypass logic during illegal conditions failing.
9.	When rd_en and fifo_empty is HIGH then error must go HIGH	PASSED

10.	Whe fifo_empty is HIGH and clear is HIGH then error must go HIGH	PASSED
11.	When fifo_empty, rd_en and clear all are HIGH then error must go HIGH	PASSED
12.	When FIFO is full and wr_en is asserted then error must go HIGH	PASSED
13.	When clear and wr_en both are asserted then error must go HIGH	PASSED
14.	When RESET is asserted and either of rd_en, wr_en or clear are asserted then error must go HIGH	PASSED
15	If clear is asserted, wrptr decreases by 1	9 errors in random testing

BLACK BOX CHECKING:

Reference model constructed using SystemVerilog queue datatype. In-built functions like .delete(), .pop_back, .pop_front etc were used to perform operations. We assumed that clear had the highest priority followed by write and then read. The checker is not cycle accurate and checks the DUT outputs at the negative edge. Most of our checking was Black Box Verification

BUGS:

- NO bugs in directed test cases.
- Unable to perform same cycle reads in some cases (Read after performing an illegal clears/writes or multiple clears).
- Rd_data sometimes not ‘bx when clear or write is asserted along with read.
- Bypassing when fifo is not empty.
- Error when performing clears in corner case conditions (like clear till fifo empty + read with clear)
- Rdptr fluctutates at times when fifo is empty and read is asserted with another control signal.
- Most errors were caused by the same cycle read bug – subsequent reads will output the previously unread data.

D. <u>COVERAGE</u>:

Inputs: RESET, clear, wr_en, rd_en, wr_data

Outputs: fifo_empty, fifo_full, rd_data, error

i. Single Coverage:

Sr. No.	**Signal**	**Coverage implemented for observing:**
1	clear	clear=0 and clear=1
2	rd_en	rd_en=0 and rd_en=1
3	wr_en	wr_en=0 and wr_en=1
4	wr_data	All values from 0 to 2^8, collected in 5 different bins.
5	rd_data	All values from 0 to 2^8, collected in 5 different bins.
6	fifo_empty	fifo_empty=0 and fifo_empty=1
7	fifo_full	fifo_full=0 and fifo_full=1
8	fifo_empty transition	Transition from 0->1 and Transition from 1->0
9	fifo_full transition	Transition from 0->1 and Transition from 1->0
10	Error	Error =0 and error =1
11	RESET	RESET = 0 and RESET =1

i. Cross Coverage:

Sr. No.	**Signal**	**Coverage implemented as:**
1	fifo_full and wr_en both are HIGH	fifo_full x wr_en
2	fifo_empty and rd_en both are HIGH	fifo_empty x rd_en
3	wr_en and clear both are HIGH	wr_en x clear
4	rd_en and clear both are HIGH	rd_en x clear
5	rd_en and wr_en both are HIGH	rd_en x wr_en
6	fifo_empty, rd_en and wr_en all are HIGH	fifo_empty x rd_en x wr_en
7	fifo_full, rd_en and wr_en all are HIGH	fifo_full x rd_en x wr_en
8	fifo_empty and rd_en are HIGH	fifo_empty x rd_en
9	rd_en , wr_en and clear are all HIGH	rd_en x wr_en x clear
10	Error, when the different control signals are asserted (reset,clear,read,write)	Error x rd_en x wr_en x clear x reset

Coverage includes single coverage of every input and output and cross coverage of 2 or more signals. Obviously, all possible combinations won't have hits in the cross-coverage bins (Example full and empty can never be 1s at the same time)

E. **SCENARIOS: (all tests are by default black box checks)**

Basic tests:

TEST	SIGNALS/COMMANDS	EXPECTED RESULT
Reset	RESET	All outputs be 'bx
Write	wr_en	FIFO_empty should be 0. rd_data should be 'x
Read	rd_en	FIFO_full should be 0. rd_data should contain the data written to the FIFO
Undo	Clear	FIFO_empty should be 0. rd_data should be 'x
Full	Reset -> 8 consecutive Writes	FIFO_empty should de-assert. FIFO_full should assert.
Empty	Full -> 8 consecutive Reads or Clears	FIFO_full should de-assert. FIFO_empty should assert.
Bypass	Empty+ Read+Write	FIFO_empty should always be 1. rd_data available in the same cycle.

Advanced tests:

TEST	COMMANDS	EXPECTED RESULT
Write when Full	Reset -> More than 9 consecutive Writes	Element not written in FIFO (wr_data ignored) and FIFO_full asserted.
Read when Empty	Full -> More than 9 consecutive Reads	Element read is 'bx and FIFO_empty asserted
Clear when Empty	Empty + Clear	Ignored
Bypass when NOT Empty	Bypass + ~Empty	Write prioritized
Clear asserted with Read	Clear + Read	Clear Operation (Insufficient data)
Clear asserted with Write	Clear + Write	Clear Operation (Insufficient data)
Clear asserted with Read and Write	Clear + Bypass	Clear Operation (Insufficient data)
Reset asserted with any combination of input signals	Reset + X	Reset
Back-to-Back Writes	Repeat (X) Write	Data Written repeatedly
Back-to-Back reads	Repeat (X) Read	ERROR!!! Last element not read

Back-to-back clear and writes	Repeat (X) Write -> Clear	Data Written and cleared repeatedly. Write pointer +1 -> -1
Corner case Write	Write 0 , F, A, 5	Data written
Clear+Write when full	Full, Clear+ Write	Clear
Clear+ Write when empty	Empty, Clear+ Write	Ignored
No operation	All control signals de asserted	Ignored (NOP)
Clear+Read when full	Full, Clear+Read	Clear
Clear+Read when empty	Empty, Clear+Read	Ignored
Reset when full	Full then reset	Reset
Reset when empty	Empty then reset	Reset

2. PROJECT MANAGEMENT

A. TOOLS:

Questa Sim, MobaXterm, GitHub, Notepad++, Google Docs, Microsoft Office.

B. RISKS & DEPENDANCIES:

- Test case errors from black box checking.
- Assertion errors from white box checking.
- Some corner cases may be left unverified.
- The results of some corner operations are still unknown/unexplored.
- The design must work for other FIFO sizes.
- Golden reference should be error free.

C. RESOURCES:

- Canvas Project description
- Questa Sim user manual
- SystemVerilog LRM

D. SCHEDULE:

WEEKS	MILESTONE
1-2	FIFO RTL Design
3-4	Bypass logic and Coverage
5-6	Testcases
7-8	Checker + Assertions

NOTE: Highlighted cases are the custom improvements Prof. Olson asked us to make!