Test Plan Document for DUT (FIFO-based OR Gate Design)

1. Design Under Test

The DUT includes 3 FIFOs (`a_ff`, `b_ff`, `y_ff`). It accepts input data through write operations to `a_ff` , `b_ff` ORs the first elements of both, and writes the result into `y_ff`.

2. Testbench Components

Component	Description
Driver	Stimulates the DUT with write/read commands and input data.
Monitor	Observes DUT outputs and sends them to the Scoreboard.
Checker	Compares expected and actual output from `y_ff`.
Scoreboard	checks whether the DUT behaves correctly by comparing expected from the monitor results with actual results from the DUT
Environment	Instantiates and connects all components.
Testcases	Specific scenarios to validate correctness of DUT.

3. Testcases

ID	Test Description	Inputs
TC1	Simple OR Test	a = 0, b = 1 y = 1
TC2	Both Inputs 0	a = 0, b = 0 y = 0
TC3	Both Inputs 1	a = 1, b = 1 y = 1
TC4	FIFO full	
TC5	FIFO empty	
TC6	Multiple write and read	

4. Corner Cases

- Writing when `a_ff` or `b_ff` is full.
- Reading from empty `y_ff`.
- Checking when no data is written at all.
- Rapid alternating write/read on same cycle.
- Invalid address .

5. Functional Coverage

Coverage Bin Name	Condition
write_a_ff	Write occurs to a_ff
write_b_ff	Write occurs to b_ff
read_y_ff	Read occurs from y_ff
or_result_0	Output of OR = 0
or_result_1	Output of OR = 1
both_inputs_1	a = 1, b = 1
both_inputs_0	a = 0, b = 0

6. Cross Coverage

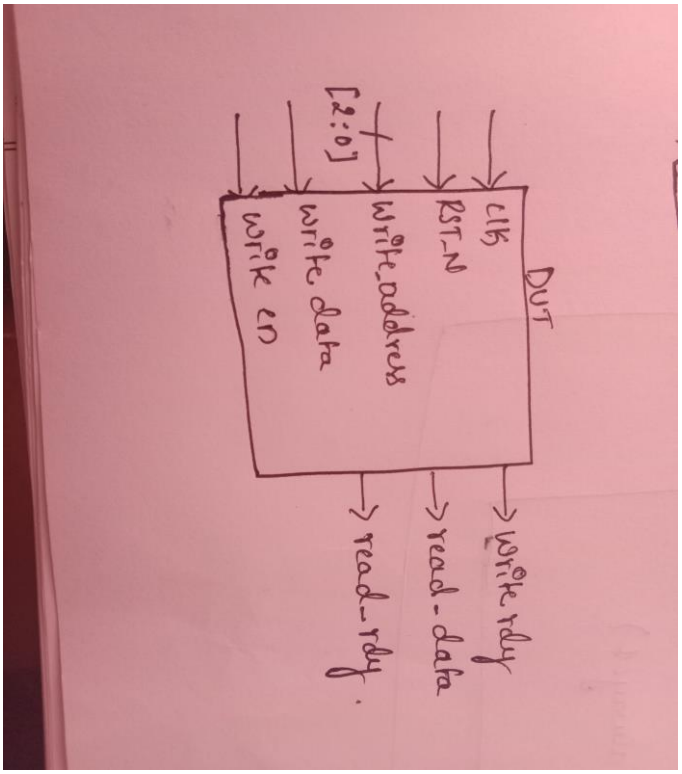
Cross Bin	Description
a_input × b_input	Check all 2×2 input combinations: 00, 01, 10, 11
write_addr × data	Check data combinations at write addr 4 and 5

7. Directory Structure

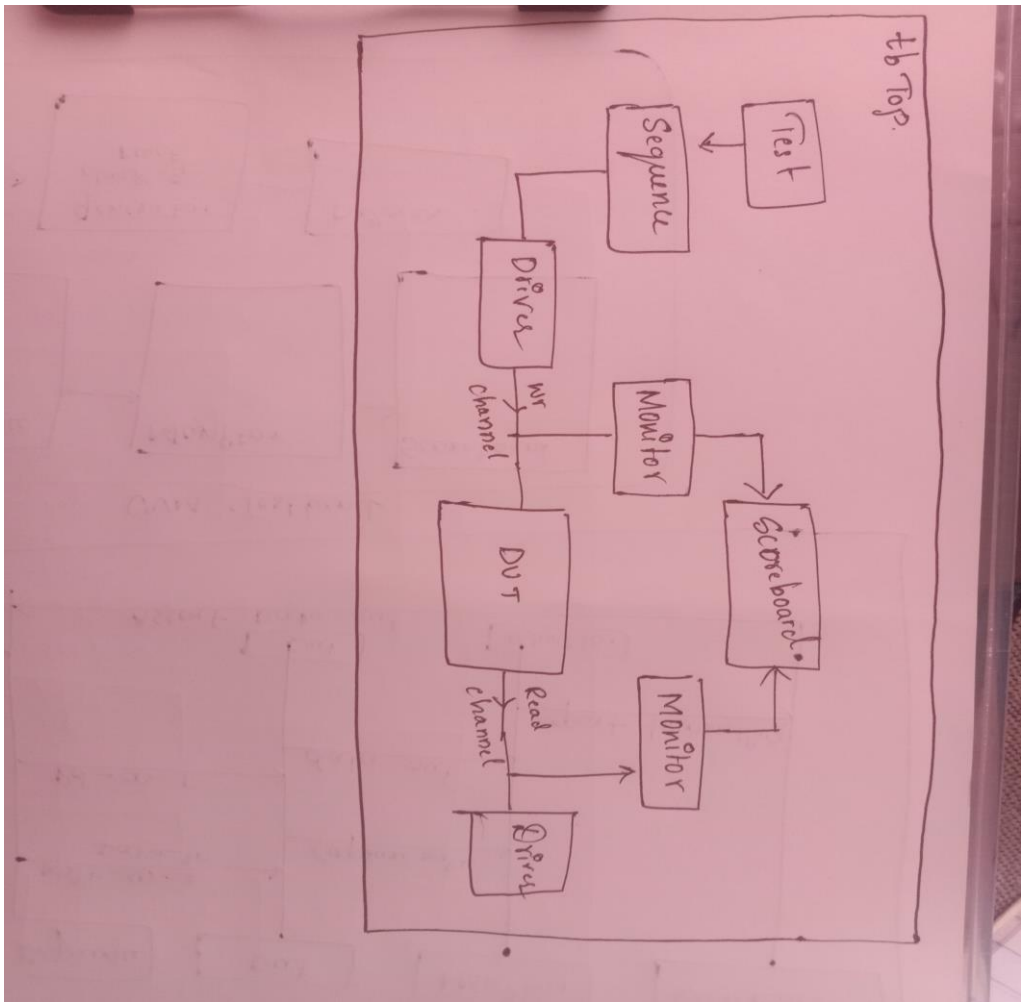
```
── bsv
        ── mem_if.bsv
── hdl
```

```
        ├── FIFO1.v
        ├── FIFO2.v
        ├── delayed_dut.v
        ├── dut.v
        ├── dut_wrapper.v
└── tests/
    ├── driver_monitor.py
    ├── dut_test.py
    ├── env.py
    ├── scoreboard.py
    └── Makefile
```

8. Design



9. Testbench


tb Top
Test
Sequence
Driver
WR
channel
Monitor
DUT
Read
channel
Monitor
Scoreboard
Driver