# System Verilog Lab2

After completing this lab, you should be able to:

*Implement function/tasks for both write and read*

*Use queues to store the write data*

*Verify DUT behaviour with waveform.*

## LAB2:

***Copy the lab1 code as lab2***

## *Step 1: Write a task read  stimulus for reading from the memory.*

1. Open testbench.sv
2. Define task with address as arguments with input direction.
3. Add the below code in Section 4 in testbench.sv

```
task read_stimulus(input reg[2:0] addr_t);
        wr<=1;
        addr<=addr_t;
        @(posedge clk);
endtask
```

## *Step 2: Add the task to initial block.*

1. Call the read methods you have implemented so far.
2. Add the below code into Section 5 in testbench.sv

```
initial begin
    reset_stimulus();

// Write task (address,wdata)
    write_stimulus(0,10);
    write_stimulus(1,20);
    write_stimulus(3,30);
    write_stimulus(4,40);

// Read task (address)
    read_stimulus(0);
    read_stimulus(1);
    read_stimulus(3);
    read_stimulus(4);

// Wait for the DUT to process the last transaction
    @(posedge clk);

    $finish;
end
```

## *Step 3: Generating and adding Golden values*

1. Declare a queue “exp_q” in Section 1.

```
reg [7:0] exp_q[$];
```

2. Add the below code in Section 4 inside the write_stimulus task in testbench.sv

```
task write_stimulus(input reg[2:0] addr_t,reg [7:0] wdata_t);
    wr<=0;
    wdata<=wdata_t;

    exp_q.push_back(wdata_t);

    addr<=addr_t;
    @(posedge clk);
endtask
```

3. Print the golden values.
4. Add the below code in Section 6 in testbench.sv

```
$display(“[Golden Values] Data = %0p”,exp_q);
```

## *Step 4: Simulating the design and Validate the output*

```
# Loading work.testbench(fast)
# Loading work.ram_dut(fast)
# [Testbench] Reset applied to DUT
# [Testbench] Reset completed
# [Golden Values] Data = 10 20 30 40
# ** Note: $finish    : testbench.sv(70)
```