**Experiment 4**

**Aim:**

Write the SystemVerilog testbench code for 256 word memory. Use the concept of System Verilog “interface”.

Memory has following ports:

Input ports: clk, rst, ce, we

Input ports: data\_in, addr\_in of 8bits

Output port: data\_out of 8-bit size.

**Part 1:** **Write the SV code for interface.**

**Code:**

interface mem\_if(

  input clk,

  input rst,

  input we,

  input ce,

  input [7:0] datain,

  input [7:0] address,

  output logic [7:0] dataout);

endinterface

**Part 2:** **Write the SV code for Memory with interface i.e. DUT with interface.**

**Code:**

module memory(mem\_if mif);

  logic [7:0] mem [0:255];

  always@(posedge mif.clk) begin

    if(mif.rst)

      mif.dataout<=8'd0;

  else if(mif.ce && !mif.we)

    mem[mif.address]<=mif.dataout;

  end

  always@(posedge mif.clk)begin

  if(mif.ce && mif.we)

    mem[mif.address]<=mif.datain;

  end

endmodule

**Part 3:** **Write the SV testbench to perform the write operation (store 10 random values in the locations with address 0, 1,2….9. Also perform the read operations to display the content written in the memory during write operation.**

**Code:**

module tb();

  logic clk =1;

  initial forever #5 clk=!clk;

  logic rst,ce,we;

  logic [7:0]datain, address;

  wire [7:0]dataout;

  mem\_if dut(clk,rst,we,ce,datain,address,dataout);

  memory uut(.mif(dut));

  initial begin

    rst<=1; ce=1'b0; we=1'b0; address<=0; datain<=0;

    repeat(10) @(negedge clk);

    rst<=0;

    for(int i=0;i<3;i=i+1) begin

      @(negedge clk) ce<=1'b1; we<=1'b1; address<=i; datain<=$random;

      @(negedge clk) ce<=1'b0;

      $display("write address %0d, data %0d",address,datain);

    end

    for(int i=0;i<3;i=i+1) begin

      @(negedge clk) ce<=1'b1; we<=1'b0; address<=i;

      $display("write address %0d,data %0d",address,uut.mif.dataout);

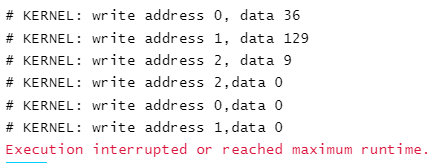
      repeat(2) @(negedge clk) ce<=1'b0;

    end

  end

endmodule

**Output:**

****

**Result:**

The given problem statement is executed and verified to be correct.