Experiment 4 – Memory and Interface

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_int(
  input wire clk,rst,ce,we,
  input wire [7:0] data_in,addr_in,
  output logic [7:0] data_out);
endinterface
```

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

Code:

```
module ram(mem_int inf);
  logic [7:0] memory [256];

  always@(posedge inf.clk) begin
    if(inf.rst) inf.data_out <= 8'd0;
    else if (inf.ce) begin
      if(inf.we) memory[inf.addr_in] <= inf.data_in;
      else if(!inf.we) inf.data_out <= memory[inf.addr_in];
      else inf.data_out <= 8'd0;
    end
  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=0,rst,ce,we;
  logic [7:0] data_in,addr_in;
  wire [7:0] data_out;

  mem_int int_tb(clk,rst,ce,we,data_in,addr_in,data_out);
  ram uut(int_tb);

  initial forever #5 clk = ~clk;

  task reset();
    begin
      rst <= 1'b1;
      ce <= 1'b0;
      repeat(2) @(posedge clk);
      rst <= 1'b0;
    end
  endtask

  task read(input [7:0] addr);
    logic [7:0] test;
    begin
      @(negedge clk);
      ce <= 1'b1;
      we <= 1'b0;
      addr_in <= addr;
      @(posedge clk) test <= uut.inf.data_out;
      @(negedge clk);
      ce <= 1'b0;
      $display("the data out is %d at %d",uut.inf.data_out,uut.inf.addr_in);
    end
  endtask

  task write(input [7:0] addr);
    begin
      @(negedge clk);
      ce <= 1'b1;
      we <= 1'b1;
      addr_in <= addr;
      data_in <= $random;
      @(negedge clk);
      ce <= 1'b0;
      $display("value written is %d at %d", uut.inf.data_in,uut.inf.addr_in);
    end
  endtask
```

```
initial begin
    reset;
    for(int i=0; i<10; i++) write(i[7:0]);
    for(int i=0; i<10; i++) read(i[7:0]);
  end
endmodule
```

Output:

```
# KERNEL: value written is  36 at   0
# KERNEL: value written is 129 at   1
# KERNEL: value written is   9 at   2
# KERNEL: value written is  99 at   3
# KERNEL: value written is  13 at   4
# KERNEL: value written is 141 at   5
# KERNEL: value written is 101 at   6
# KERNEL: value written is  18 at   7
# KERNEL: value written is   1 at   8
# KERNEL: value written is  13 at   9
# KERNEL: the data out is  36 at   0
# KERNEL: the data out is 129 at   1
# KERNEL: the data out is   9 at   2
# KERNEL: the data out is  99 at   3
# KERNEL: the data out is  13 at   4
# KERNEL: the data out is 141 at   5
# KERNEL: the data out is 101 at   6
# KERNEL: the data out is  18 at   7
# KERNEL: the data out is   1 at   8
# KERNEL: the data out is  13 at   9
```

Result:

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