VIDYAVARDHAKA COLLEGE OF ENGINEERING

Gokulam III Stage, Mysuru – 570 002

Department of Electronics and Communication Engineering

IV Semester

SystemVerilog Lab Manual
21EC43L

Prepared by

Alfred Vivek D'Souza
Assistant Professor
Department of ECE, VVCE

1. Course Overview

This laboratory course enables the students to understand concepts related to verification of digital systems described using Hardware Description Language (HDL) with System Verilog. This course teaches them to write testbenches for automated verification for different types of digital systems.

2. Syllabus

1. Verification of Combinational Circuits using layered testbench architecture
    a. Half Adder
    b. 4 Line to 1 Line Multiplexer
    c. 4 bit Magnitude Comparator
2. Verification of 16 bit signed Full Adder using layered testbench architecture
3. Verification of 32 bit ALU using layered testbench architecture with constrained random stimuli
4. Verification of Systems using Data Structures
    a. Verification of RAM using Arrays

3. Prerequisites of the course

1. Digital System Design
2. Hardware Description Language

4. Course Objectives

The objectives of this laboratory course are:

1. Understand the use of testbenches for automated verification of digital systems.
2. Understand layered testbench architecture and constrained random testing
3. Using System Verilog for writing testbenches to verify the digital systems.

5. Course Outcomes

Upon completion of the course, students will be able to

1. Verify the digital systems by implementing layered testbenches using System Verilog.

6. CO-PO-PSO Mapping

Course Outcomes	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO1	PSO2

CO1				3									2	

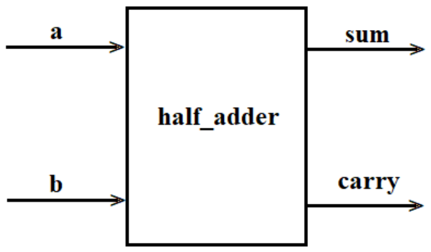
Contents

1. Verification of combinational circuits using layered test bench architecture

1.1. Half Adder

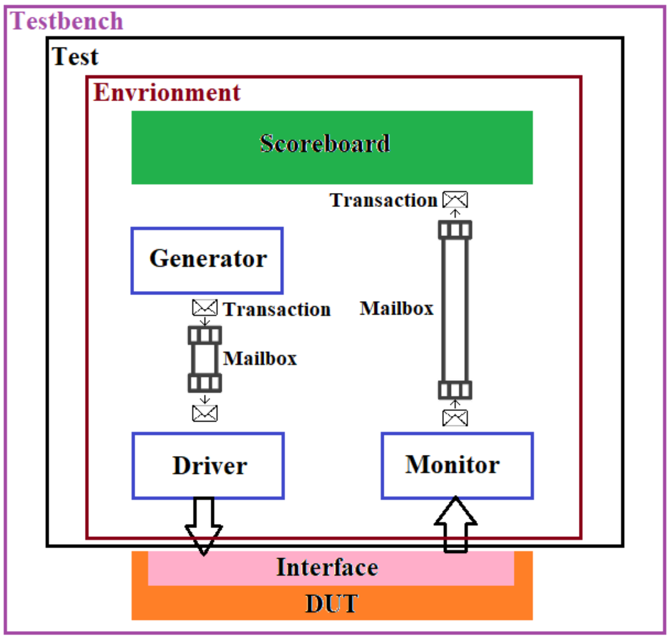
1.1.1. Description

DUT	:	half_adder
DUT description	:	1 bit half adder
Design file(s)	:	half_adder_i1.vp, half_adder_i2.vp, half_adder_i3.vp, half_adder_i4.vp



Signal description	:	a and b are input signals of bit type sum and carry are output signals of bit type
Functionality	:	half_adder is a pure combinational design sum is asserted when a and b are dissimilar ie., $sum = a \land b$ carry is asserted when both a and b are logic high ie., $carry = a \& b$

1.1.2. Test bench Architecture



1.1.3. Test bench Code

File Name : Interface.sv
Details : This file contains definition of interface named intf
Contents :

```
interface intf();
  bit a;
  bit b;
  bit sum;
  bit carry;
endinterface
```

File Name : Transaction.sv
Details : This file contains implementation of a class called transaction.
Contents :

```
class transaction;

  rand bit a;
  rand bit b;
  bit sum;
  bit carry;

  function void display(string name);
    $display("In: %s: a = %0d, b = %0d, sum = %0d, carry = %0d",name,
a,b,sum,carry);
    $display("-------------------------------------");
  endfunction

endclass
```

File Name : Generator.sv
Details : This file contains implementation of a class called generator.
Contents :

```
class generator;

  transaction trans;
  mailbox gen2driv;

  function new(mailbox gen2driv);
    this.gen2driv = gen2driv;
  endfunction
  task run();
    repeat(1)
      begin
        trans = new();
        trans.randomize();
        trans.display("Generator");
        gen2driv.put(trans);
      end
  endtask

endclass
```

File Name : Driver.sv

Details : This file contains implementation of a class called driver.

Contents :

```
class driver;

  virtual intf vif;
  mailbox gen2driv;

  function new(virtual intf vif, mailbox gen2driv);
    this.vif = vif;
    this.gen2driv = gen2driv;
  endfunction

  task run();
    repeat(1)
      begin
        transaction trans;
        gen2driv.get(trans);

        vif.a <= trans.a;
        vif.b <= trans.b;

        trans.sum = vif.sum;
        trans.carry = vif.carry;
        trans.display("Driver");

      end
  endtask

endclass
```

File Name	:	Monitor.sv
Details	:	This file contains implementation of a class called monitor.
Contents	:	

```
class monitor;

  virtual intf vif;
  mailbox mon2scb;

  function new(virtual intf vif, mailbox mon2scb);
    this.vif = vif;
    this.mon2scb = mon2scb;
  endfunction

  task run();
    repeat(1)
      #3;
      begin
          transaction trans;
          trans = new ();
          trans.a = vif.a;
          trans.b = vif.b;
          trans.sum = vif.sum;
          trans.carry = vif.carry;
          mon2scb.put(trans);
          trans.display("Monitor");
      end
  endtask

endclass
```

File Name	:	Scoreboard.sv
Details	:	This file contains implementation of a class called scoreboard.
Contents	:	

```
class scoreboard;
  mailbox mon2scb;

  function new(mailbox mon2scb);
    this.mon2scb = mon2scb;
  endfunction

  task run();
    transaction trans;
    repeat(1)
      begin
        mon2scb.get(trans);
        if(((trans.a ^ trans.b) == trans.sum) && ((trans.a & trans.b)
== trans.carry))
          $display("Correct");
        else
          $display("Wrong");

        trans.display("Scoreboard");
      end
  endtask

endclass
```

File Name	:	Environment.sv
Details	:	This file contains implementation of a class called environment which is a container for all the components of the testbench.
Contents	:	

```
`include "Transaction.sv"
`include "Generator.sv"
`include "Driver.sv"
`include "Monitor.sv"
`include "Scoreboard.sv"

class environment;
  generator gen;
  driver driv;
  monitor mon;
  scoreboard scb;
  mailbox m1, m2;
  virtual intf vif;

  function new(virtual intf vif);
    this.vif = vif;
    m1 = new();
    m2 = new();
    gen = new(m1);
    driv = new(vif, m1);
    mon = new(vif, m2);
    scb = new(m2);
  endfunction

  task test();
    fork
      gen.run();
      driv.run();
      mon.run();
      scb.run();
    join
  endtask
  task run();
    test();
  endtask

endclass
```

File Name	:	Test.sv
Details	:	This file contains implementation of a class called test which controls the simulation environment.
Contents	:	

```
`include "Environment.sv"

program test(intf i_intf);
  environment env;
  initial
    begin
      env = new(i_intf);
      env.run();
      env.run();
      env.run();
      $finish;
    end
endprogram
```

File Name	:	Testbench.sv
Details	:	This file contains top module named tbench_top which connects interface and the instance of DUT.
Contents	:	

```
`include "Interface.sv"
`include "Test.sv"

module tbench_top;

  intf i_intf();
  test t1(i_intf);

  half_adder h1
(.a(i_intf.a),.b(i_intf.b),.sum(i_intf.sum),.carry(i_intf.carry));

endmodule
```

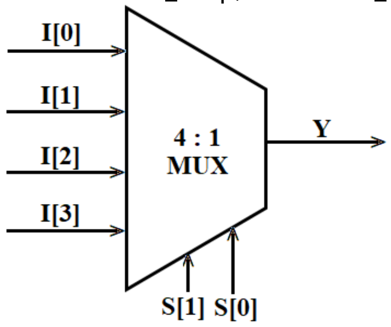
1.1.4. Introduced Errors

Sl.No	Encrypted source code file	Errors	Source code file
1.	half_adder_i1.vp	Error in carry	half_adder_i1.v
2.	half_adder_i2.vp	None	half_adder_i2.v
3.	half_adder_i3.vp	Error in sum. Sum is always first operand.	half_adder_i3.v
4.	half_adder_i4.vp	Error in sum. Sum is always zero.	half_adder_i4.v

1.2. 4 Line to 1 Line Multiplexer

1.2.1. Description

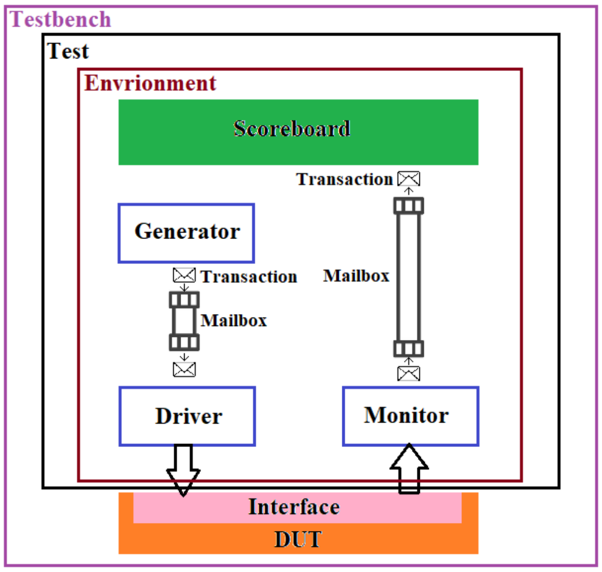
DUT	:	MUX4to1
DUT description	:	4 to 1 line data selector or multiplexer
Design file	:	MUX4To1_i1.vp, MUX4To1_i2.vp, MUX4To1_i3.vp



Signal description	:	I[3:0] are input channels, Y is the output and S[1:0] are select lines All the signals are of bit type
Functionality	:	MUX4To1 is a pure combinational design Operation is shown in truthtable below

S[1]	S[0]	Y
0	0	I[0]
0	1	I[1]
1	0	I[2]
1	1	I[3]

1.2.2. Test bench Architecture



1.2.3. Test bench Code

File Name	:	Interface.sv
Details	:	This file contains definition of interface named intf
Contents	:	

```
interface intf();
bit [3:0]I;
bit [1:0]S;
         bit Y;
endinterface
```

File Name	:	Transaction.sv
Details	:	This file contains implementation of a class called transaction.
Contents	:	

```
class transaction;

  randc bit [3:0]I;
  randc bit [1:0]S;
  bit Y;

  function void display(string name);
    $display("In: %s: I = %b, S = %b, Y = %b",name, I, S, Y);
    $display("-------------------------------------");
  endfunction

endclass
```

File Name	:	Generator.sv
Details	:	This file contains implementation of a class called generator.
Contents	:	

```
class generator;
  transaction trans;
  mailbox gen2driv;

  function new(mailbox gen2driv);
    this.gen2driv = gen2driv;
  endfunction

  task run();
    repeat(1)
```

```
      begin
        trans = new();
        trans.randomize();
        trans.display("Generator");
        gen2driv.put(trans);
      end
  endtask
endclass
```

File Name	:	Driver.sv
Details	:	This file contains implementation of a class called driver.
Contents	:	

```
class driver;

  mailbox gen2driv;
  transaction trans;
  virtual intf vif;

  function new(virtual intf vif, mailbox gen2driv);
    this.vif = vif;
    this.gen2driv = gen2driv;
  endfunction

  task run();
    repeat(1)
      begin
        gen2driv.get(trans);

        vif.I <= trans.I;
        vif.S <= trans.S;

        trans.Y = vif.Y;
        trans.display("Driver");

      end
  endtask

endclass
```

File Name	:	Monitor.sv
Details	:	This file contains implementation of a class called monitor.
Contents	:	

```
class monitor;

  virtual intf vif;
  mailbox mon2scb;

  function new(virtual intf vif, mailbox mon2scb);
    this.vif = vif;
    this.mon2scb = mon2scb;
  endfunction

  task run();
    repeat(1)
      #3;
      begin
          transaction trans;
          trans = new ();
          trans.I = vif.I;
          trans.S = vif.S;
          trans.Y = vif.Y;
          mon2scb.put(trans);
          trans.display("Monitor");
      end
  endtask

endclass
```

File Name	:	Scoreboard.sv
Details	:	This file contains implementation of a class called scoreboard.
Contents	:	

```
class scoreboard;
  mailbox mon2scb;
  transaction trans;
  function new(mailbox mon2scb);
    this.mon2scb = mon2scb;
  endfunction

  task run();

    repeat(1)
      begin
        mon2scb.get(trans);
        if(trans.S == 2'b00 && trans.Y == trans.I[0])
          $display("Correct");
        else if(trans.S == 2'b01 && trans.Y == trans.I[1])
          $display("Correct");
        else if(trans.S == 2'b10 && trans.Y == trans.I[2])
```

```
          $display("Correct");
        else if(trans.S == 2'b11 && trans.Y == trans.I[3])
          $display("Correct");
        else
          $display("Wrong");

        trans.display("Scoreboard");
      end
  endtask

endclass
```

File Name	:	Environment.sv
Details	:	This file contains implementation of a class called environment which is a container for all the components of the testbench.
Contents	:	

```
`include "Transaction.sv"
`include "Generator.sv"
`include "Driver.sv"
`include "Monitor.sv"
`include "Scoreboard.sv"

class environment;
  generator gen;
  driver driv;
  monitor mon;
  scoreboard scb;
  mailbox m1, m2;
  virtual intf vif;

  function new(virtual intf vif);
    this.vif = vif;
    m1 = new();
    m2 = new();
    gen = new(m1);
    driv = new(vif, m1);
    mon = new(vif, m2);
    scb = new(m2);
  endfunction

  task test();
    fork
      gen.run();
      driv.run();
      mon.run();
      scb.run();
    join
  endask
  task run();
    test();
  endtask

endclass
```

File Name	:	Test.sv
Details	:	This file contains implementation of a class called test which controls the simulation environment.
Contents	:	

```
`include "Environment.sv"

program test(intf i_intf);
  environment env;
  initial
    begin
      repeat(16)
      begin
           env = new(i_intf);
           env.run();
      end
      $finish;
    end
endprogram
```

File Name	:	Testbench.sv
Details	:	This file contains top module named tbench_top which connects interface and the instance of DUT.
Contents	:	

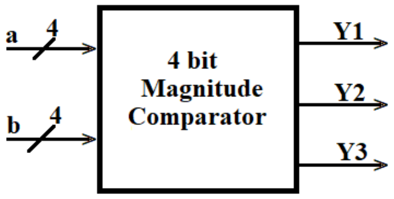
```
`include "Interface.sv"
`include "Test.sv"

module tbench_top;

  intf i_intf();
  test t1(i_intf);
  MUX_4To1 M1(.Y(i_intf.Y),.I(i_intf.I),.S(i_intf.S));

endmodule
```

1.2.4. Introduced Errors

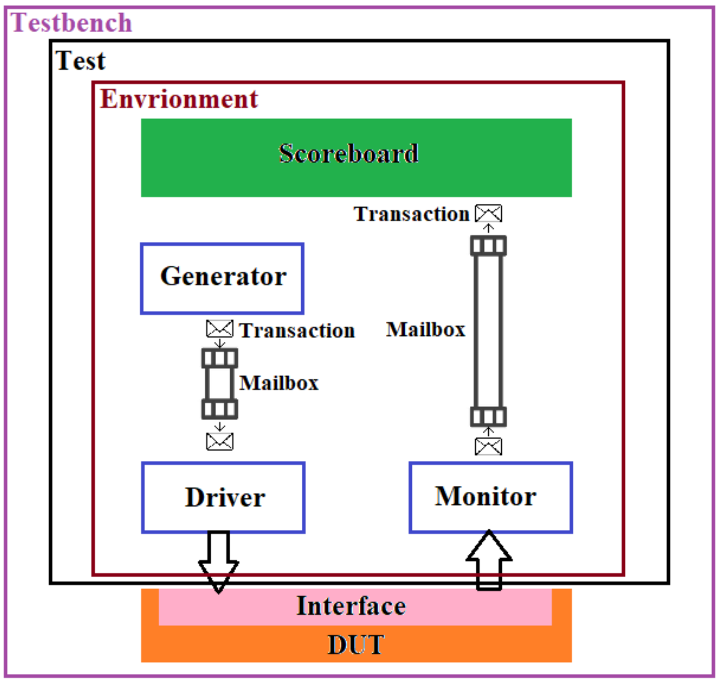
Sl.No	Encrypted source code file	Errors	Source code file
1.	MUX4To1_i1.vp	No Errors	MUX4To1_i1.v
2.	MUX4To1_i2.vp	S = 01 and S = 10 exchanged	MUX4To1_i2.v
3.	MUX4To1_i3.vp	Outputs are inverted	MUX4To1_i3.v

1.3. 4 bit Magnitude Comparator

1.3.1. Description

DUT	:	mag_comparator_4bit
DUT description	:	4 bit magnitude comparator
Design File	:	mag_comparator_4bit.vp
		
Signal description	:	All signals are of bit type a and b are 4-bit inputs Y1, Y2 and Y3 are 1-bit outputs
Functionality	:	mag_comparator_4bit is a pure combinational design Y1 is high if a > b Y2 is high if a = b Y3 is high if a < b

1.3.2. Test bench Architecture



1.3.3. Test bench Code

File Name	:	Interface.sv
Details	:	This file contains definition of interface named intf
Contents	:	

```
interface intf();
  bit [3:0]a;
  bit [3:0]b;
  bit Y1,Y2,Y3;
endinterface
```

File Name	:	Transaction.sv
Details	:	This file contains implementation of a class called transaction.
Contents	:	

```
class transaction;

  randc bit [3:0]a;
  randc bit [3:0]b;
  bit Y1, Y2, Y3;

  function void display(string name);
    $display("In: %s: a = %b, b = %b, Y1 = %b, Y2 = %b, Y3 =
%b",name, a, b, Y1, Y2, Y3);
    $display("-------------------------------------");
  endfunction

endclass
```

File Name	:	Generator.sv
Details	:	This file contains implementation of a class called generator.
Contents	:	

```
class generator;
  transaction trans;
  mailbox gen2driv;

  function new(mailbox gen2driv);
    this.gen2driv = gen2driv;
  endfunction

  task run();
    repeat(1)
      begin
        trans = new();
        trans.randomize();
        trans.display("Generator");
        gen2driv.put(trans);
      end
  endtask

endclass
```

File Name	:	Driver.sv
Details	:	This file contains implementation of a class called driver.
Contents	:	

```
class driver;

  mailbox gen2driv;
  virtual intf vif;

  function new(virtual intf vif, mailbox gen2driv);
    this.vif = vif;
    this.gen2driv = gen2driv;
  endfunction

  task run();
    repeat(1)
      begin
        transaction trans;
        gen2driv.get(trans);

        vif.a <= trans.a;
        vif.b <= trans.b;

        trans.Y1 = vif.Y1;
        trans.Y2 = vif.Y2;
        trans.Y3 = vif.Y3;
        trans.display("Driver");

      end
  endtask

endclass
```

File Name	:	Monitor.sv
Details	:	This file contains implementation of a class called monitor.
Contents	:	

```
class monitor;

  virtual intf vif;
  mailbox mon2scb;
  transaction trans;

  function new(virtual intf vif, mailbox mon2scb);
    this.vif = vif;
    this.mon2scb = mon2scb;
  endfunction

  task run();
    repeat(1)
      #3;
      begin
          trans = new ();
          trans.a = vif.a;
          trans.b = vif.b;
          trans.Y1 = vif.Y1;
          trans.Y2 = vif.Y2;
          trans.Y3 = vif.Y3;
          mon2scb.put(trans);
          trans.display("Monitor");
      end
  endtask

endclass
```

File Name	:	Scoreboard.sv
Details	:	This file contains implementation of a class called scoreboard.
Contents	:	

```
class scoreboard;
  mailbox mon2scb;

  function new(mailbox mon2scb);
    this.mon2scb = mon2scb;
  endfunction

  task run();
    transaction trans;
    repeat(1)
      begin
        mon2scb.get(trans);
        if(trans.a > trans.b && trans.Y1 == 1 && trans.Y2 == 0 &&
trans.Y3 == 0)
          $display("Correct");
        else if (trans.a == trans.b && trans.Y1 == 0 && trans.Y2 == 1
&& trans.Y3 == 0)
          $display("Correct");
        else if (trans.a < trans.b && trans.Y1 == 0 && trans.Y2 == 0
&& trans.Y3 == 1)
```

```
          $display("Correct");
        else
          $display("Wrong");

        trans.display("Scoreboard");
      end
  endtask

endclass
```

File Name	:	Environment.sv
Details	:	This file contains implementation of a class called environment which is a container for all the components of the testbench.
Contents	:	

```
`include "Transaction.sv"
`include "Generator.sv"
`include "Driver.sv"
`include "Monitor.sv"
`include "Scoreboard.sv"

class environment;
  generator gen;
  driver driv;
  monitor mon;
  scoreboard scb;
  mailbox m1, m2;
  virtual intf vif;

  function new(virtual intf vif);
    this.vif = vif;
    m1 = new();
    m2 = new();
    gen = new(m1);
    driv = new(vif, m1);
    mon = new(vif, m2);
    scb = new(m2);
  endfunction

  task test();
    fork
      gen.run();
      driv.run();
      mon.run();
      scb.run();
    join
  endask

  task run();
```

```
    test();
  endtask

endclass
```

File Name	:	Test.sv
Details	:	This file contains implementation of a class called test which controls the simulation environment.

Contents :

```
`include "Environment.sv"

program test(intf i_intf);
  environment env;
  initial
    begin
      repeat(256)
      begin
           env = new(i_intf);
           env.run();
      end
      $finish;
    end
endprogram
```

File Name	:	Testbench.sv
Details	:	This file contains top module named tbench_top which connects interface and the instance of DUT.

Contents :

```
`include "Interface.sv"
`include "Test.sv"

module tbench_top;

  intf i_intf();
  test t1(i_intf);
  mag_comparator_4bit
M1(.a(i_intf.a),.b(i_intf.b),.Y1(i_intf.Y1),.Y2(i_intf.Y2),.Y3(i_intf
.Y3));

endmodule
```

1.3.4. Introduced Errors

Sl.No	Encrypted source code file	Errors	Source code file
1.	mag_comparator_4bit_i1.vp	No Errors	mag_comparator_4bit_i1.v
2.	mag_comparator_4bit_i2.vp	Greater than and Lesser than are exchanged	mag_comparator_4bit_i2.v

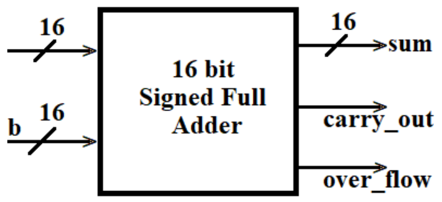
2. Verification of 16 bit signed Full Adder using layered testbench architecture

The 16 bit full adder uses 2's compliment form to represent negative numbers. The MSB acts as the sign bit. The number should be treated as positive if the sign bit is 0 else negative. The range of 16 bit signed number is from – 32768 to 32767 (-2^{N-1} to $2^{N-1} - 1$ where N = 16). After performing addition if the result exceeds the range, then overflow is said to have occurred. Overflow never occurs if Addend and Augend are of different sign but occurs if

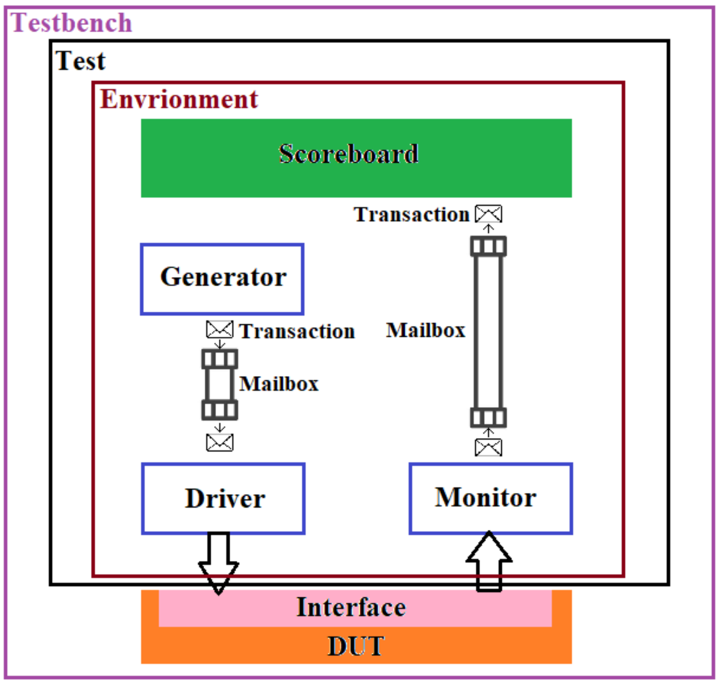
- Both Addend and Augend are negative and the sum is less than – 32768.
- Both Addend and Augend are positive and the sum is greater than 32767.

Adding two 16 bit unsigned numbers using the Full Adder results in a 17 bit sum. The 17th bit of the sum is the carry out of the Full Adder.

2.1.1. Description

DUT	:	full_adder_16bit
DUT description	:	Signed 16 bit full adder
Design File	:	Full_adder_16bit.vp
		 16 b 16 16 bit Signed Full Adder 16 sum carry_out over_flow
Signal description	:	a and b are 16 bit inputs representing augend and addend respectively sum is 16 bit output carry_out and over_flow are 1 bit output
Functionality	:	full_adder_16bit is a pure combinational design a and b are signed numbers represented in 2's complement form sum = a + b carry_out is asserted if the sum needs more than 16 bit over_flow is asserted if the sum overflows

2.1.2. Test bench Architecture



2.1.3. Test bench Code

File Name	:	Interface.sv
Details	:	This file contains definition of interface named intf
Contents	:	

```
interface intf();
      shortint a;
      shortint b;
      shortint sum;
      bit carry_out;
      bit over_flow;
endinterface
```

File Name	:	Transaction.sv
Details	:	This file contains implementation of a class called transaction.
Contents	:	

```
class transaction;
      rand shortint a;
      rand shortint b;
      shortint sum;
      bit carry_out;
      bit over_flow;

      function display(string name);
            $display("In: %s, a = %0d, b = %0d, sum = %0d, carry_out =
%b, over_flow = %b", name, a, b, sum, carry_out, over_flow);
            $display("In: %s, a = %b, b = %b, sum = %b, carry_out =
%b, over_flow = %b", name, a, b, sum, carry_out, over_flow);
      endfunction
endclass
```

File Name	:	Generator.sv
Details	:	This file contains implementation of a class called generator.
Contents	:	

```
class generator;
      transaction t1;
      mailbox gen2driv;

      function new(mailbox gen2driv);
            this.gen2driv = gen2driv;
      endfunction

      task run();
            t1 = new();
            t1.randomize();
            t1.display("Generator");
            gen2driv.put(t1);
      endtask
endclass
```

File Name	:	Driver.sv
Details	:	This file contains implementation of a class called driver.
Contents	:	

```
class driver;
      transaction t1;
      mailbox gen2driv;
      virtual intf vif;

      function new(virtual intf vif, mailbox gen2driv);
            this.vif = vif;
            this.gen2driv = gen2driv;
      endfunction

      task run();
            gen2driv.get(t1);
            vif.a <= t1.a;
            vif.b <= t1.b;
            t1.sum = vif.sum;
            t1.carry_out = vif.carry_out;
            t1.over_flow = vif.over_flow;
            t1.display("Driver");
      endtask
endclass
```

File Name	:	Monitor.sv
Details	:	This file contains implementation of a class called monitor.
Contents	:	

```
class monitor;
      transaction t1;
      mailbox mon2scb;
      virtual intf vif;

      function new(virtual intf vif, mailbox mon2scb);
            this.vif = vif;
            this.mon2scb = mon2scb;
      endfunction

      task run();
            #3;
            t1 = new();
            t1.a = vif.a;
            t1.b = vif.b;
            t1.sum = vif.sum;
            t1.carry_out = vif.carry_out;
            t1.over_flow = vif.over_flow;
            t1.display("Monitor");
            mon2scb.put(t1);
      endtask
endclass
```

File Name	:	Scoreboard.sv
Details	:	This file contains implementation of a class called scoreboard.

Contents :

```
class scoreboard;
      transaction t1;
      mailbox mon2scb;

      function new(mailbox mon2scb);
            this.mon2scb = mon2scb;
      endfunction

      task run();
            mon2scb.get(t1);
            /* Over flow detection */
            if(t1.a < 0 && t1.b < 0 && (t1.a + t1.b) < -32768)
            begin
                  if(t1.over_flow == 1'b1)
                        $display("Correct");
                  else
                        $display("Wrong");
            end
            else if (t1.a > 0 && t1.b > 0 && (t1.a + t1.b) > 32767)
            begin
                  if(t1.over_flow == 1'b1)
                        $display("Correct");
                  else
                        $display("Wrong");
            end
            else if((t1.a + t1.b) == t1.sum)
                  $display("Correct");
            else
                  $display("Wrong");

            t1.display("Scoreboard");
      endtask

endclass
```

File Name	:	Environment.sv
Details	:	This file contains implementation of a class called environment which is a container for all the components of the testbench.

Contents :

```
`include "Transaction.sv"
`include "Generator.sv"
`include "Driver.sv"
`include "Monitor.sv"
`include "Scoreboard.sv"
```

```
class environment;
  generator gen;
  driver driv;
  monitor mon;
  scoreboard scb;
  mailbox m1, m2;
  virtual intf vif;

  function new(virtual intf vif);
    this.vif = vif;
    m1 = new();
    m2 = new();
    gen = new(m1);
    driv = new(vif, m1);
    mon = new(vif, m2);
    scb = new(m2);
  endfunction

  task test();
    fork
      gen.run();
      driv.run();
      mon.run();
      scb.run();
    join
  endask

  task run();
    test();
  endtask

endclass
```

File Name	:	Test.sv
Details	:	This file contains implementation of a class called test which controls the simulation environment.
Contents	:	

```
`include "Environment.sv"

program test(intf i_intf);
  environment env;
  initial
    begin
      repeat(256)
      begin
           env = new(i_intf);
           env.run();
      end
      $finish;
    end
endprogram
```

File Name	:	Testbench.sv
Details	:	This file contains top module named tbench_top which connects interface and the instance of DUT.
Contents	:	

```
`include "interface.sv"
`include "test.sv"

module tbench_top;

      intf i_intf();
      test t1(i_intf);

      full_adder_16bit f1
(.a(i_intf.a),.b(i_intf.b),.sum(i_intf.sum),.carry_out(i_intf.carry_o
ut),.over_flow(i_intf.over_flow));

endmodule
```

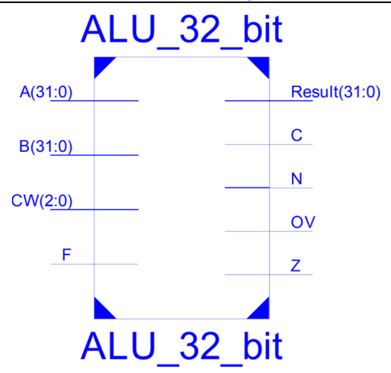
2.1.4. Introduced Errors

Sl.No	Encrypted source code file	Errors	Source code file
1.	Full_adder_16bit_i1.vp	Overflow logic is incorrect	Full_adder_16bit_i1.v
2.	Full_adder_16bit_i2.vp	Performs subtraction instead of addition	Full_adder_16bit_i2.v
3.	Full_adder_16bit_i3.vp	No Errors	Full_adder_16bit_i3.v

3. Verification of 32 bit ALU using layered testbench architecture with constrained random stimuli

3.1. 32 bit ALU verification demo

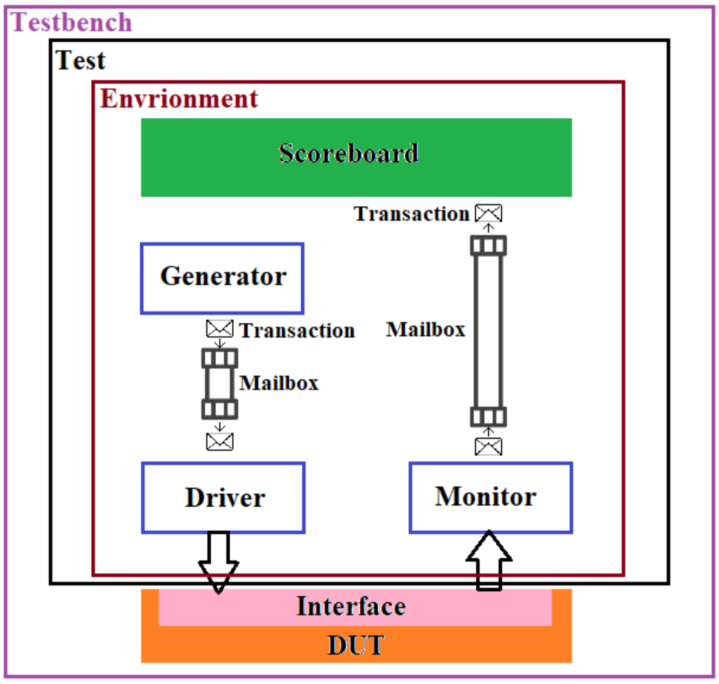
3.1.1. Description

DUT	:	ALU_32_bit
DUT description	:	32 bit Arithmetic and Logic Unit
Design File	:	ALU_32_bit_i1.vp, ALU_32_bit_i2.vp, ALU_32_bit_i3.vp, ALU_32_bit_i4.vp
 ALU_32_bit A(31:0) B(31:0) CW(2:0) F Result(31:0) C N OV Z ALU_32_bit		
Signal description	:	A, B are 32 bit inputs CW is 3 bit input F is 1 bit input Result is 32 bit output C, N, OV and Z are 1 bit outputs
Functionality	:	ALU_32_bit is a pure combinational design A, B and Result must be treated as signed numbers for arithmetic operations F is used to select one of the functions: arithmetic or logic and CW is used to select operation C is carry/borrow flag and becomes high if there is a carry/borrow during arithmetic operations N is negative flag and becomes high if the result is negative OV is overflow flag and becomes high if results overflow during arithmetic operations Z is zero flag and becomes high if result is zero The functionality is given in the table below.

ALU Functions:

F	CW	Operation	Flags to Consider			
			C	N	OV	Z
0 (Logical Functions)	000	Result = A AND B	X	X	X	✓
	001	Result = A OR B	X	X	X	✓
	010	Result = A NAND B	X	X	X	✓
	011	Result = A EXOR B	X	X	X	✓
	100	Result = NOT A	X	X	X	✓
	101, 110, 111	**Invalid**				
1 (Arithmetic Functions)	000	Result = A + B	✓	✓	✓	✓
	001	Result = A + 1	✓	✓	✓	✓
	010	Result = A - 1	✓	✓	✓	✓
	011, 100, 101,110,111	**Invalid**				

3.1.2. Test bench Architecture



3.1.3. Test bench Code

File Name	:	Interface.sv
Details	:	This file contains definition of interface named intf
Contents	:	

```
interface intf();
      int A, B, Result;
      bit [2:0] CW;
      bit F, C, N, OV, Z;
endinterface
```

File Name	:	Transaction.sv
Details	:	This file contains implementation of a class called transaction.
Contents	:	

```
class transaction;

      randc int A, B;
      randc bit [2:0] CW;
      randc bit F;
      int Result;
      bit C, N, OV, Z;

      constraint op_select
      {
            (F == 1'b0) -> CW inside {[0:4]};
            (F == 1'b1) -> CW inside {[0:2]};
      }

      function display(string name);
            $display("In: %s, A = %0d, B = %0d, F = %b, CW = %b, 
Result = %d, C = %b, N = %b, OV = %b, Z = %b", name, A, B, F, CW, 
Result, C, N, OV, Z);
            //$display("In: %s, A = %b, B = %b, F = %b, CW = %b, 
Result = %b, C = %b, N = %b, OV = %b, Z = %b", name, A, B, F, CW, 
Result, C, N, OV, Z);
      endfunction
endclass
```

File Name	:	Generator.sv
Details	:	This file contains implementation of a class called generator.
Contents	:	

```
class generator;
      transaction t1;
      mailbox gen2driv;

      function new(mailbox gen2driv);
            this.gen2driv = gen2driv;
      endfunction

      task run();
            t1 = new();
            t1.randomize();
            t1.display("Generator");
            gen2driv.put(t1);
      endtask
endclass
```

File Name	:	Driver.sv
Details	:	This file contains implementation of a class called driver.
Contents	:	

```
class driver;
      transaction t1;
      mailbox gen2driv;
      virtual intf vif;

      function new(virtual intf vif, mailbox gen2driv);
            this.vif = vif;
            this.gen2driv = gen2driv;
      endfunction

      task run();
            gen2driv.get(t1);
            vif.A <= t1.A;
            vif.B <= t1.B;
            vif.F <= t1.F;
            vif.CW <= t1.CW;
            t1.Result = vif.Result;
            t1.C = vif.C;
            t1.N = vif.N;
            t1.OV = vif.OV;
            t1.Z = vif.Z;
            t1.display("Driver");
      endtask
endclass
```

File Name	:	Monitor.sv
Details	:	This file contains implementation of a class called monitor.

Contents :

```
class monitor;
      transaction t1;
      mailbox mon2scb;
      virtual intf vif;

      function new(virtual intf vif, mailbox mon2scb);
            this.vif = vif;
            this.mon2scb = mon2scb;
      endfunction

      task run();
            #3;
            t1 = new();
            t1.A = vif.A;
            t1.B = vif.B;
            t1.F = vif.F;
            t1.CW = vif.CW;
            t1.Result = vif.Result;
            t1.C = vif.C;
            t1.N = vif.N;
            t1.OV = vif.OV;
            t1.Z = vif.Z;
            t1.display("Monitor");
            mon2scb.put(t1);
      endtask
endclass
```

File Name	:	Scoreboard.sv
Details	:	This file contains implementation of a class called scoreboard.

Contents :

```
class scoreboard;
      transaction t1;
      mailbox mon2scb;

      function new(mailbox mon2scb);
            this.mon2scb = mon2scb;
      endfunction

      task run();
            longint signed result;
            bit carryFlag, negFlag, overflowFlag, zeroFlag;

            mon2scb.get(t1);

            if(t1.F == 0)
            begin

                  case(t1.CW)
                        3'b000: result = t1.A & t1.B;
                        3'b001: result = t1.A | t1.B;
                        3'b010: result = ~(t1.A & t1.B);
                        3'b011: result = t1.A ^ t1.B;
                        3'b100: result = ~t1.A;
                        default: begin
                                    $display("Invalid Test Case");
                                    return;
                               end
                  endcase
                  if(result == 0)
                        zeroFlag = 1;
                  else
                        zeroFlag = 0;

                  if(result == t1.Result && zeroFlag == t1.Z)
                        $display("Passed");
                  else
                        $display("Failed");
            end
            else if(t1.F == 1)
            begin
                  case(t1.CW)
                        3'b000: result = t1.A + t1.B;
                        3'b001: result = t1.A + 1;
                        3'b010: result = t1.A - 1;
                        default: begin
                                    $display("Invalid Test Case");
                               end
                  endcase

                  if(result >= -2.147483648e9 && result <=
2.147483647e9 )
                  begin
                        if(result == t1.Result)
                        begin
                              negFlag = (result < 0)? 1 : 0;
                              zeroFlag = (result == 0) ? 1 : 0;
                              if(negFlag == t1.N && zeroFlag == t1.Z)
                                    $display("Passed");
                              else
                                    $display("Failed: Flags");
                        end
                        else
                              $display("Failed");
                  end
                  else if(t1.OV == 1 && int'(result) == t1.Result)
                        $display("Passed: Overflow");
                  else
```

```
                        $display("Failed");

            end
            else
                  $display("Invalid Test Case");

            t1.display("Scoreboard");
      endtask

endclass
```

File Name	:	Environment.sv
Details	:	This file contains implementation of a class called environment which is a container for all the components of the testbench.
Contents	:	

```
`include "transaction.sv"
`include "generator.sv"
`include "driver.sv"
`include "monitor.sv"
`include "scoreboard.sv"

class environment;
      generator gen;
      driver driv;
      monitor mon;
      scoreboard scb;
      mailbox m1, m2;
      virtual intf vif;

      function new(virtual intf vif);
            this.vif = vif;
            m1 = new();
            m2 = new();
            gen = new(m1);
            driv = new(vif, m1);
            mon = new(vif, m2);
            scb = new(m2);
      endfunction

      task test();
      fork
                  gen.run();
                  driv.run();
                  mon.run();
                  scb.run();
      join
      endtask
      task run();
            test();
      endtask
endclass
```

File Name	:	Test.sv
Details	:	This file contains implementation of a class called test which controls the simulation environment.
Contents	:	

```
`include "environment.sv"
program test(intf i_intf);
      environment env;
      initial
      begin
                  env = new(i_intf);
                repeat(25)
                begin
                      env.run();
                      $display("--------------------");
                end
                  $finish;
      end
endprogram
```

File Name	:	Testbench.sv
Details	:	This file contains top module named tbench_top which connects interface and the instance of DUT.
Contents	:	

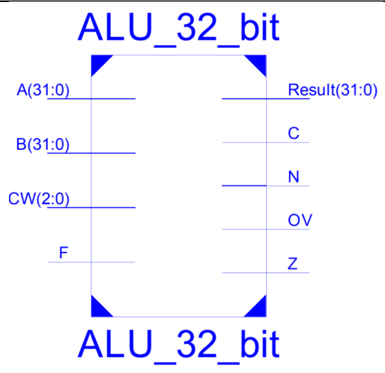
```
`include "interface.sv"
`include "test.sv"

module tbench_top;

      intf i_intf();
      test t1(i_intf);

      ALU_32_bit A1 (  .A(i_intf.A), .B(i_intf.B),
                  .CW(i_intf.CW), .F(i_intf.F),
                  .Result(i_intf.Result),
                  .Z(i_intf.Z), .N(i_intf.N),
                  .C(i_intf.C), .OV(i_intf.OV)
                     );
endmodule
```

3.1.4. Introduced Errors

Sl.No	Encrypted source code file	Errors	Source code file
1.	ALU_32_bit_i1.vp	No Errors	ALU_32_bit_i1.v
2.	ALU_32_bit_i2.vp	NOT operation on B instead of A	ALU_32_bit_i2.v
3.	ALU_32_bit_i3.vp	INC and DEC operations does opposite	ALU_32_bit_i3.v
4.	ALU_32_bit_i4.vp	N and Z flags are always equal	ALU_32_bit_i4.v

3.2. 32 bit ALU verification exercise

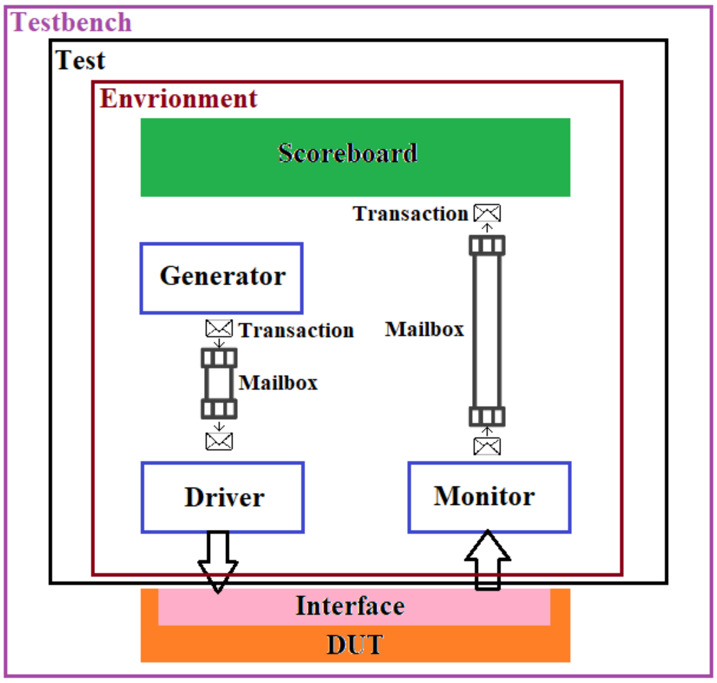
3.2.1. Description

DUT	:	ALU_32_bit
DUT description	:	32 bit Arithmetic and Logic Unit
Design File	:	ALU_32_bit_i1.vp, ALU_32_bit_i2.vp, ALU_32_bit_i3.vp, ALU_32_bit_i4.vp
 ALU_32_bit A(31:0) B(31:0) CW(2:0) F Result(31:0) C N OV Z ALU_32_bit		
Signal description	:	A, B are 32 bit inputs CW is 3 bit input F is 1 bit input Result is 32 bit output C, N, OV and Z are 1 bit outputs
Functionality	:	ALU_32_bit is a pure combinational design A, B and Result must be treated as signed numbers for arithmetic operations F is used to select one of the functions: arithmetic or logic and CW is used to select operation C is carry/borrow flag and becomes high if there is a carry/borrow during arithmetic operations N is negative flag and becomes high if the result is negative OV is overflow flag and becomes high if results overflow during arithmetic operations Z is zero flag and becomes high if result is zero The functionality is given in the table below.

ALU Functions:

F	CW	Operation	Flags to Consider			
			C	N	OV	Z
0 (Logical Functions)	000	Result = A AND B	X	X	X	✓
	001	Result = A OR B	X	X	X	✓
	010	Result = A NAND B	X	X	X	✓
	011	Result = A NOR B	X	X	X	✓
	100	Result = NOT A	X	X	X	✓
	101, 110, 111	**Invalid**				
1 (Arithmetic Functions)	000	Result = A + B	✓	✓	✓	✓
	001	Result = A – B	✓	✓	✓	✓
	010	Result = A + 1	✓	✓	✓	✓
	011	Result = A - 1	✓	✓	✓	✓
	100, 101,110,111	**Invalid**				

3.2.2. Test bench Architecture



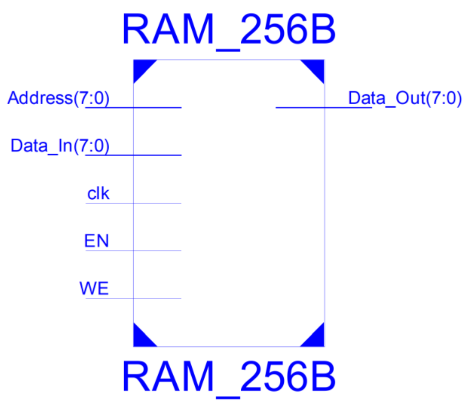
3.2.3. Introduced Errors

Sl.No	Encrypted source code file	Errors	Source code file
1.	Exercise_ALU_32_bit_i1.vp	A+B is not performed instead A – B is performed	Exercise_ALU_32_bit_i1.v
2.	Exercise_ALU_32_bit_i2.vp	No Errors	Exercise_ALU_32_bit_i2.v
3.	Exercise_ALU_32_bit_i3.vp	EXOR is performed instead of OR	Exercise_ALU_32_bit_i3.v

4. Verification of Systems using Data Structures

4.1. Verification of 256 Bytes RAM using Array

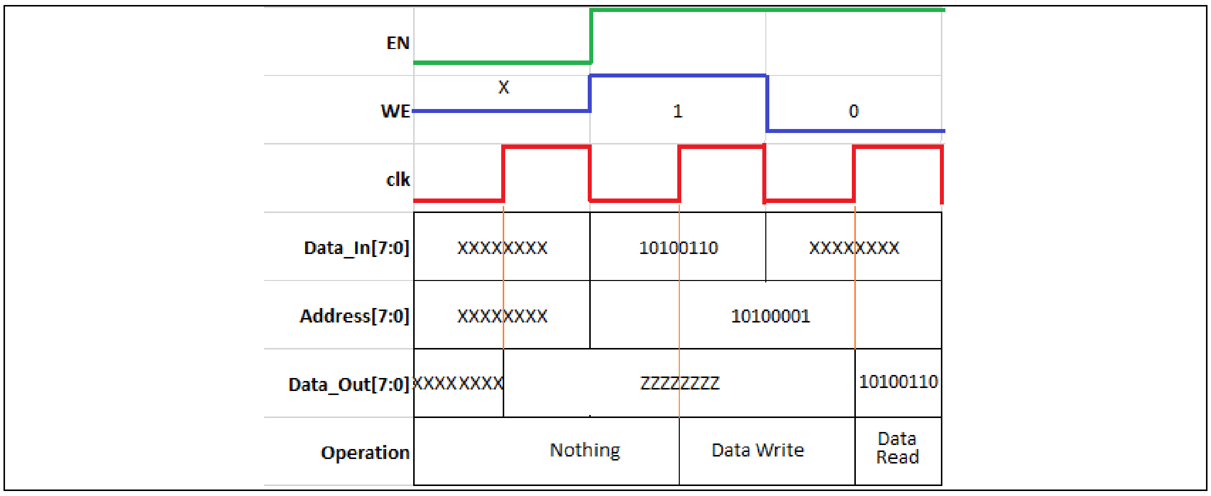
4.1.1. Description

DUT	:	RAM_256B
DUT description	:	256 bytes Single Port Random Access Memory
Design File	:	RAM_256B_i1.vp
 RAM_256B Address(7:0) Data_In(7:0) clk EN WE Data_Out(7:0) RAM_256B		
Signal description	:	Address and Data_In are 8 bit inputs clk, EN and WE are 1 bit inputs Data_Out is 8 bit output
Functionality	:	Write and Read operations are synchronous with positive edge of clk signal. The functionality is given in the table below.

EN	WE	Operations
0	X	Data_Out is high impedance
1	1	Write to memory: Data_In will be stored in specified address.
1	0	Read from memory: Data_Out will have data read from specified address.

The EN, WE, Data_In and Address signals must be stable at the time of arrival of positive edge of clock

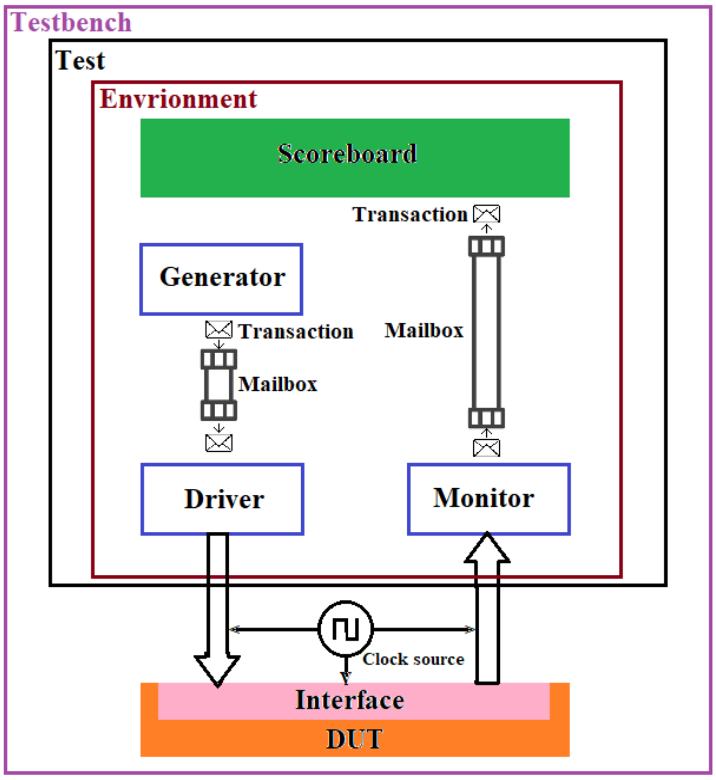
Typical timing diagram is shown below



NOTE:

1. All the memory locations must be written to and read from at least once with EN = 1 or EN = 0.
2. EN = 0 must not exceed 10% of test cases.

4.1.2. Test bench Architecture



4.1.3. Test bench Code

File Name	:	Interface.sv
Details	:	This file contains definition of interface named intf
Contents	:	

```
interface intf();
	byte unsigned Data_In;
	logic [7:0] Data_Out;
	byte unsigned Address;
	bit clk, EN, WE;
endinterface
```

File Name	:	Transaction.sv
Details	:	This file contains implementation of a class called transaction.
Contents	:	

```
class transaction;

	rand byte unsigned Data_In;
	rand bit EN;
	byte unsigned Address;
	bit WE;
	logic [7:0] Data_Out;

	bit [8:0] temp = 9'd0;

	constraint EN_dist {
		EN dist {0 := 1, 1 := 9};
	};

	function RandomGen();
		this.randomize();
		Address = temp[7 : 0];
		WE = ~temp[8];
		temp++;
	endfunction

	function display(string name);
		$display("In: %s, Data_In = %h, Address = %h, En = %b, WE
= %b, Data_Out = %h  ",name, Data_In, Address, EN, WE, Data_Out);

	endfunction

	function transaction copy();
		copy = new();
		copy.Data_In = this.Data_In;
```

```
            copy.Address = this.Address;
            copy.EN = this.EN;
            copy.WE = this.WE;
            copy.Data_Out = this.Data_Out;
      endfunction;
endclass
```

File Name	:	Generator.sv
Details	:	This file contains implementation of a class called generator.
Contents	:	

```
class generator;
      transaction t1;
      mailbox gen2driv;
      int transaction_count;
      event ended;

      function new(mailbox gen2driv, event ended);
            this.gen2driv = gen2driv;
            this.ended = ended;
            t1 = new();
      endfunction

      task run();
            repeat(transaction_count)
            begin
                  t1.RandomGen();
                  t1.display("Generator");
                  gen2driv.put(t1.copy());
                  #20;
            end
            ->ended;
      endtask
endclass
```

File Name	:	Driver.sv
Details	:	This file contains implementation of a class called driver.
Contents	:	

```
class driver;
      transaction t1;
      mailbox gen2driv;
      virtual intf vif;
      int transaction_count = 0;

      function new(virtual intf vif, mailbox gen2driv);
            this.vif = vif;
            this.gen2driv = gen2driv;
      endfunction

      task run();
      forever
         begin
            gen2driv.get(t1);
            vif.Data_In <= t1.Data_In;
            vif.Address <= t1.Address;
            vif.EN <= t1.EN;
            vif.WE <= t1.WE;
            t1.display("Driver");
            @(posedge vif.clk);
            transaction_count++;
         end
      endtask
endclass
```

File Name	:	Monitor.sv
Details	:	This file contains implementation of a class called monitor.
Contents	:	

```
class monitor;
      transaction t1;
      mailbox mon2scb;
      virtual intf vif;

      function new(virtual intf vif, mailbox mon2scb);
            this.vif = vif;
            this.mon2scb = mon2scb;
      endfunction

      task run();
            t1 = new();
            forever
            begin
                  @(posedge vif.clk);
                  #5;
                  t1.Data_In = vif.Data_In;
                  t1.Address = vif.Address;
                  t1.EN = vif.EN;
                  t1.WE = vif.WE;
                  t1.Data_Out = vif.Data_Out;
                  t1.display("Monitor");
                  mon2scb.put(t1.copy());
            end
      endtask
endclass
```

File Name	:	Scoreboard.sv
Details	:	This file contains implementation of a class called scoreboard.

Contents :

```
class scoreboard;
      transaction t1;
      mailbox mon2scb;
      int transaction_count = 0;
      byte unsigned local_memory[256];

      function new(mailbox mon2scb);
                  int i = 0;
            this.mon2scb = mon2scb;
                  for(i = 0; i < 256; i = i + 1)
                  local_memory[i] = 8'hFF;
      endfunction

      task run();
            byte unsigned address;
            forever
            begin
                  mon2scb.get(t1);
                  address = unsigned'(t1.Address);
                  if(t1.EN == 1'b1 && t1.WE == 1'b1)
                  begin
                        local_memory[address] = t1.Data_In;
                              $display("PASS - Data Write: Saved %h in
location %h",t1.Data_In, address);
                  end
                  else if(t1.EN == 1'b1 && t1.WE == 1'b0 && t1.Data_Out
=== local_memory[address])
                        $display("PASS - Data Read : Location - %h,
Retrieved %h , Actual = %h",address, t1.Data_Out,
local_memory[address]);
                  else if(t1.EN == 1'b1 && t1.WE == 1'b0 && t1.Data_Out
!== local_memory[address])
                        $display("FAIL - Data Read : Location - %h,
Retrieved %h , Actual = %h",address, t1.Data_Out,
local_memory[address]);
                  else if(t1.EN == 1'b0 && t1.Data_Out === 8'bzzzzzzzz)
                        $display("PASS - EN = %b, Data_Out = %b",t1.EN,
t1.Data_Out);
                  else if(t1.EN == 1'b0 && t1.Data_Out !== 8'bzzzzzzzz)
                        $display("FAIL - EN = %b, Data_Out = %b",t1.EN,
t1.Data_Out);
                  else
                        $display("Some other error");

                  transaction_count++;
            end
      endtask
endclass
```

File Name	:	Environment.sv
Details	:	This file contains implementation of a class called environment which is a container for all the components of the testbench.
Contents	:	

```
`include "Transaction.sv"
`include "Generator.sv"
`include "Driver.sv"
`include "Monitor.sv"
`include "Scoreboard.sv"

class environment;
     generator gen;
     driver driv;
     monitor mon;
     scoreboard scb;
     mailbox m1, m2;
     virtual intf vif;
     event gen_ended;

     function new(virtual intf vif, int transaction_count);
          this.vif = vif;
          m1 = new();
          m2 = new();
               gen = new(m1, gen_ended);
               gen.transaction_count = transaction_count;

          driv = new(vif, m1);
          mon = new(vif, m2);
           scb = new(m2);
     endfunction

     task test();
     fork
               gen.run();
               driv.run();
          mon.run();
          scb.run();
       join_any
     endtask

     task run();
          test();
          wait(gen_ended.triggered);
          wait(gen.transaction_count == driv.transaction_count);
          wait(gen.transaction_count == scb.transaction_count);
          $finish;
```

```
      endtask
endclass
```

File Name	:	Test.sv
Details	:	This file contains implementation of a class called test which controls the simulation environment.
Contents	:	

```
`include "Environment.sv"

program test(intf i_intf);
      environment env;
      initial
      begin
           env = new(i_intf, 512);
            env.run();
      end
endprogram
```

File Name	:	Testbench.sv
Details	:	This file contains top module named tbench_top which connects interface and the instance of DUT.
Contents	:	

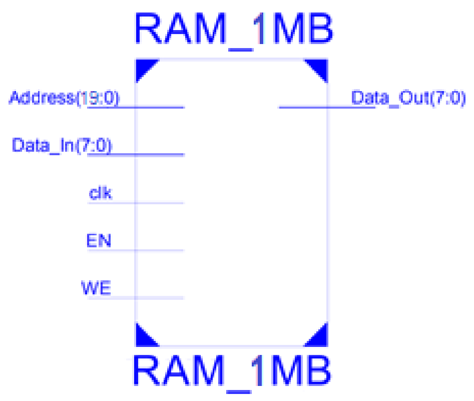
```
`include "Interface.sv"
`include "Test.sv"

module tbench_top;
      intf i_intf();
      test t1(i_intf);
      RAM_256B R1 (
            .Data_In(i_intf.Data_In),
            .Address(i_intf.Address),
            .clk(i_intf.clk),
            .EN(i_intf.EN),
            .WE(i_intf.WE),
            .Data_Out(i_intf.Data_Out)
);
      initial
      begin
            i_intf.clk <= 0;
      end

      always #10 i_intf.clk <= ~i_intf.clk;

endmodule
```

4.1.4. Introduced Errors

Sl.No	Encrypted source code file	Errors	Source code file
1.	RAM_256B_i1.vp	No Errors	RAM_256B_i1.v

4.2. Verification of 1 MB RAM using Associative Array

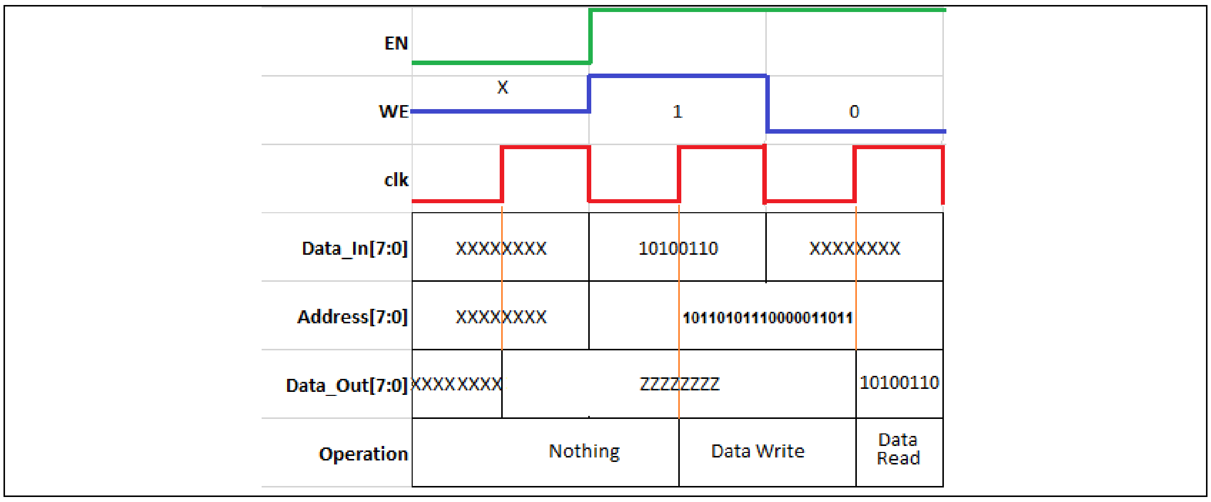
4.2.1. Description

DUT	:	RAM_1MB
DUT description	:	1 MB Single Port Random Access Memory
Design File	:	RAM_1MB_i1.vp
 RAM_1MB Address(19:0) Data_In(7:0) clk EN WE Data_Out(7:0) RAM_1MB		
Signal description	:	Data_In is 8 bit input Address is 20 bit input clk, EN and WE are 1 bit inputs Data_Out is 8 bit output
Functionality	:	Write and Read operations are synchronous with positive edge of clk signal. The functionality is given in the table below.

EN	WE	Operations
0	X	Data_Out is high impedance
1	1	Write to memory: Data_In will be stored in specified address.
1	0	Read from memory: Data_Out will have data read from specified address.

The EN, WE, Data_In and Address signals must be stable at the time of arrival of positive edge of clock

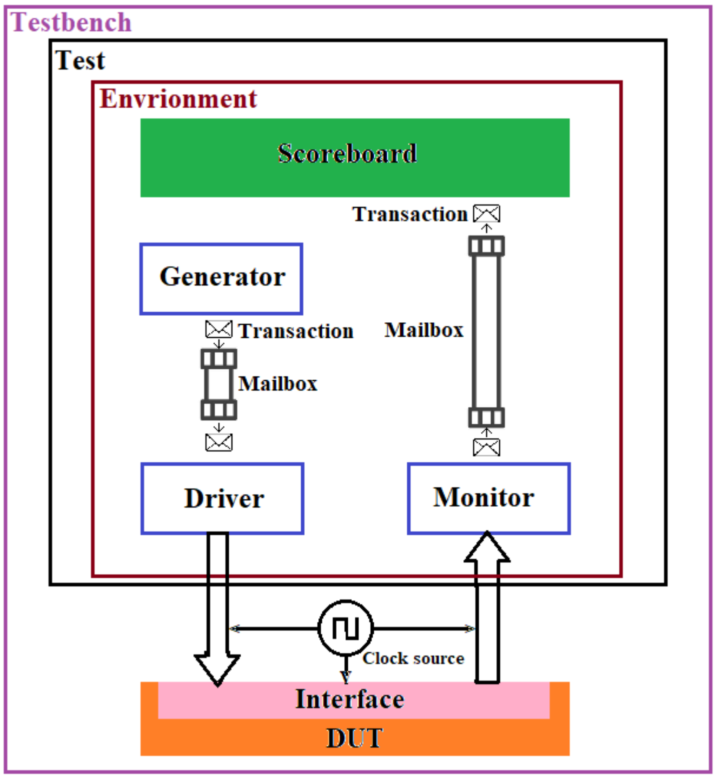
Typical timing diagram is shown below



NOTE:

1. EN = 0 must not exceed more than 10% of test cases.
2. Should verify memory write, memory read, memory write and then read from same address.

4.2.2. Test bench Architecture



4.2.3. Test bench Code

File Name	:	Interface.sv
Details	:	This file contains definition of interface named intf
Contents	:	

```
interface intf();
      byte unsigned Data_In;
      logic [7:0] Data_Out;
      bit [19:0] Address;
      bit clk, EN, WE;
endinterface
```

File Name	:	Transaction.sv
Details	:	This file contains implementation of a class called transaction.
Contents	:	

```
class transaction;

      rand byte unsigned Data_In;
      rand bit EN;
      rand bit [19:0] Address;
      rand bit WE;
      logic [7:0] Data_Out;

      rand bit duplicate;

      constraint EN_dist {
            EN dist {0 := 1, 1 := 9};
      };

      function display(string name);
            $display("In: %s, Data_In = %h, Address = %h, En = %b, WE
= %b, Data_Out = %h  ",name, Data_In, Address, EN, WE, Data_Out);

      endfunction

      function transaction copy();
            copy = new();
            copy.Data_In = this.Data_In;
            copy.Address = this.Address;
            copy.EN = this.EN;
            copy.WE = this.WE;
            copy.Data_Out = this.Data_Out;
      endfunction;
endclass
```

File Name	:	Generator.sv
Details	:	This file contains implementation of a class called generator.
Contents	:	

```
class generator;
      transaction t1;
      mailbox gen2driv;
      int transaction_count;
      event ended;

      function new(mailbox gen2driv, event ended);
            this.gen2driv = gen2driv;
            this.ended = ended;
            t1 = new();
      endfunction

      task run();
            for(int i = 0; i < transaction_count; i++)
            begin
                        t1.randomize();
                        //t1.display("Generator");
                        gen2driv.put(t1.copy());
                        #20;
                        if(t1.duplicate == 1'b1 && t1.EN == 1'b1 &&
t1.WE == 1'b1)
                        begin
                              i++;
                              t1.WE = ~t1.WE;
                              gen2driv.put(t1.copy());
                              #20;
                        end
            end
            ->ended;
      endtask
endclass
```

File Name	:	Driver.sv
Details	:	This file contains implementation of a class called driver.
Contents	:	

```
class driver;
      transaction t1;
      mailbox gen2driv;
      virtual intf vif;
      int transaction_count = 0;

      function new(virtual intf vif, mailbox gen2driv);
            this.vif = vif;
            this.gen2driv = gen2driv;
      endfunction

      task run();
            forever
                  begin
                        gen2driv.get(t1);
                        vif.Data_In <= t1.Data_In;
                        vif.Address <= t1.Address;
                        vif.EN <= t1.EN;
                        vif.WE <= t1.WE;
                        //t1.display("Driver");
                        @(posedge vif.clk);
                        transaction_count++;
                  end
      endtask
endclass
```

File Name	:	Monitor.sv
Details	:	This file contains implementation of a class called monitor.
Contents	:	

```
class monitor;
      transaction t1;
      mailbox mon2scb;
      virtual intf vif;

      function new(virtual intf vif, mailbox mon2scb);
            this.vif = vif;
            this.mon2scb = mon2scb;
      endfunction

      task run();
            t1 = new();
            forever
            begin
                  @(posedge vif.clk);
                  #5;
                  t1.Data_In = vif.Data_In;
                  t1.Address = vif.Address;
                  t1.EN = vif.EN;
                  t1.WE = vif.WE;
                  t1.Data_Out = vif.Data_Out;
                  //t1.display("Monitor");
                  mon2scb.put(t1.copy());
            end
      endtask
endclass
```

File Name	:	Scoreboard.sv
Details	:	This file contains implementation of a class called scoreboard.
Contents	:	

```
class scoreboard;
      transaction t1;
      mailbox mon2scb;
      int transaction_count = 0;
      byte unsigned local_memory[longint];

      function new(mailbox mon2scb);
            this.mon2scb = mon2scb;
      endfunction

      task run();
            byte unsigned data;
            forever
            begin
                  mon2scb.get(t1);
                  if(local_memory.exists(t1.Address))
                  data = local_memory[t1.Address];
                              else
                                 data = 8'b11111111;

                  if(t1.EN == 1'b1 && t1.WE == 1'b1)
                  begin
                                   local_memory[t1.Address] =
t1.Data_In;
                                   $display("PASS - Data Write: Saved
%h in location %h",t1.Data_In, t1.Address);
                  end
                        else if(t1.EN == 1'b1 && t1.WE == 1'b0 &&
t1.Data_Out === data)
                                   $display("PASS - Data Read :
Location - %h, Retrieved %h , Actual = %h",t1.Address, t1.Data_Out,
data);
                        else if(t1.EN == 1'b1 && t1.WE == 1'b0 &&
t1.Data_Out !== data)
                                   $display("FAIL - Data Read :
Location - %h, Retrieved %h , Actual = %h",t1.Address, t1.Data_Out,
data);
                        else if(t1.EN == 1'b0 && t1.Data_Out ===
8'bzzzzzzzz)
                        $display("PASS - EN = %b, Data_Out = %b",t1.EN,
t1.Data_Out);
                  else if(t1.EN == 1'b0 && t1.Data_Out !== 8'bzzzzzzzz)
                        $display("FAIL - EN = %b, Data_Out = %b",t1.EN,
t1.Data_Out);
```

```
                else
                     $display("Some other error");
                transaction_count++;
           end
     endtask
endclass
```

File Name	:	Environment.sv
Details	:	This file contains implementation of a class called environment which is a container for all the components of the testbench.
Contents	:	

```
`include "Transaction.sv"
`include "Generator.sv"
`include "Driver.sv"
`include "Monitor.sv"
`include "Scoreboard.sv"

class environment;
     generator gen;
     driver driv;
     monitor mon;
     scoreboard scb;
     mailbox m1, m2;
     virtual intf vif;
     event gen_ended;

     function new(virtual intf vif, int transaction_count);
           this.vif = vif;
           m1 = new();
           m2 = new();
                gen = new(m1, gen_ended);
                gen.transaction_count = transaction_count;

           driv = new(vif, m1);
           mon = new(vif, m2);
                     scb = new(m2);
     endfunction

     task test();
     fork
                gen.run();
                driv.run();
                mon.run();
                scb.run();
           join_any
     endtask

     task run();
```

```
           test();
                 wait(gen_ended.triggered);
                 wait(gen.transaction_count ==
driv.transaction_count);
                 wait(gen.transaction_count == scb.transaction_count);
                 $finish;
      endtask
endclass
```

File Name	:	Test.sv
Details	:	This file contains implementation of a class called test which controls the simulation environment.
Contents	:	

```
`include "Environment.sv"

program test(intf i_intf);
      environment env;
      initial
      begin
           env = new(i_intf, 25);
           env.run();
           $finish;
      end
endprogram
```

File Name	:	Testbench.sv
Details	:	This file contains top module named tbench_top which connects interface and the instance of DUT.
Contents	:	

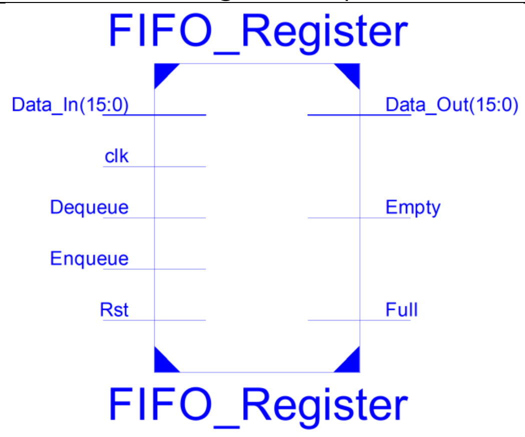
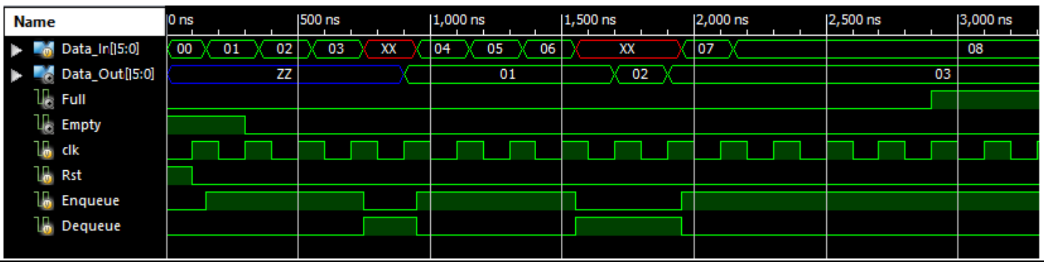
```
`include "Interface.sv"
`include "Test.sv"

module tbench_top;


      intf i_intf();
      test t1(i_intf);

      RAM_1MB R1 (
           .Data_In(i_intf.Data_In),
           .Address(i_intf.Address),
           .clk(i_intf.clk),
           .EN(i_intf.EN),
           .WE(i_intf.WE),
           .Data_Out(i_intf.Data_Out)
);

      initial
      begin
           i_intf.clk <= 0;
      end

      always #10 i_intf.clk <= ~i_intf.clk;

endmodule
```

4.2.4. Introduced Errors

Sl.No	Encrypted source code file	Errors	Source code file
1.	RAM_1MB_i1.vp	No Errors	RAM_1MB_i1.v

4.3. Verification of FIFO Register using Queue

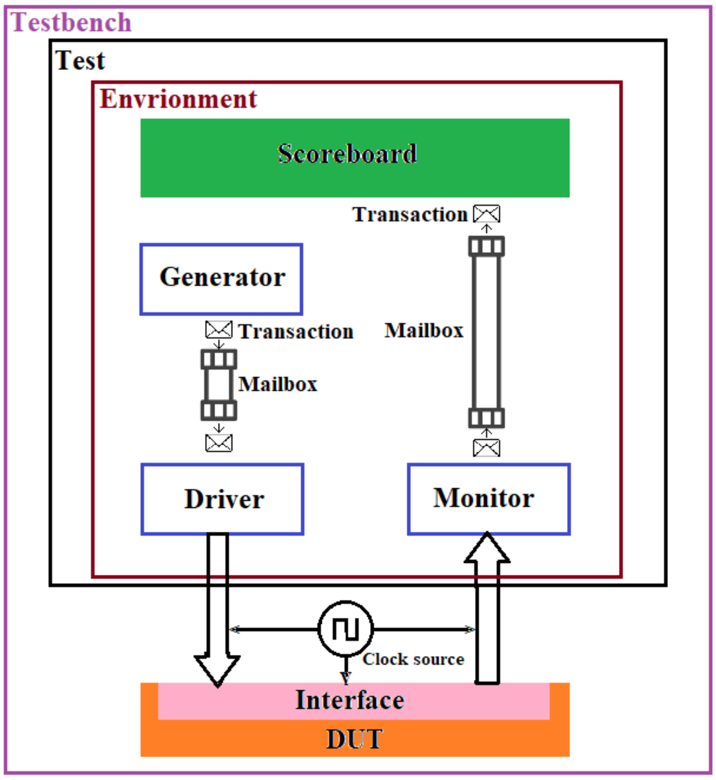
4.3.1. Description

DUT	:	FIFO_Register
DUT description	:	16 bit 8 memory location FIFO
Design File	:	FIFO_Register_i1.vp
 FIFO_Register Data_In(15:0) clk Dequeue Enqueue Rst Data_Out(15:0) Empty Full FIFO_Register		
Signal description	:	Data_In is a 16 bit input. clk, Rst, Enqueue, Dequeue are 1 bit input signals. Data_Out is a 16 bit output. Empty and Full are 1 bit output signals.
Functionality	:	Enqueue and Dequeue are synchronous operations with positive edge of clk signal. FIFO_Register is a First In First Out Buffer. It can store upto 8 data. Data can be written into FIFO_Buffer by setting Enqueue to logic 1 and passing data through Data_In. Data can be read from FIFO_Buffer by setting Dequeue to logic high. Both Enqueue and Dequeue cannot be set to logic high at the same time. Full becomes logic high if buffer is full and under such circumstances, no more data can be enqueued. Empty becomes high if buffer has no data and under such circumstances, data cannot be dequeued.
Typical timing diagram is shown below 		

NOTE:

1. Rst = 1 must not exceed 10% of test cases.

4.3.2. Test bench Architecture



4.3.3. Test bench Code

File Name	:	Interface.sv
Details	:	This file contains definition of interface named intf
Contents	:	

```
interface intf();
      bit clk, Rst, Enqueue, Dequeue, Empty, Full;
      bit [15:0] Data_In;
      logic unsigned [15:0] Data_Out;
endinterface
```

File Name	:	Transaction.sv
Details	:	This file contains implementation of a class called transaction.

Contents :

```
class transaction;

      rand bit Rst;
      rand bit Enqueue;
      rand bit [15:0] Data_In;
      bit Dequeue;
      bit Empty, Full;
      logic unsigned [15:0] Data_Out;

      constraint Rst_C {
            Rst dist { 0 := 9, 1 := 0};
      };

      function RandomGen();
            this.randomize();
            this.Dequeue = !this.Enqueue;
      endfunction

      function Display(string name);
            $display("In: %s, Data_In = %h, Data_Out = %h, Rst = %b,
Enque = %b, Dequeue = %b, Empty = %b, Full = %b",name, Data_In,
Data_Out, Rst, Enqueue, Dequeue, Empty, Full);
      endfunction

      function transaction copy();
            copy = new();
            copy.Rst = this.Rst;
            copy.Enqueue = this.Enqueue;
            copy.Data_In = this.Data_In;
            copy.Dequeue = this.Dequeue;
            copy.Empty = this.Empty;
                  copy.Full = this.Full;
            copy.Data_Out = this.Data_Out;
      endfunction;
endclass
```

File Name	:	Generator.sv
Details	:	This file contains implementation of a class called generator.
Contents	:	

```
 class generator;
      transaction t1;
      mailbox gen2driv;
      int transaction_count;
      event ended;

      function new(mailbox gen2driv, event ended);
```

```
            this.gen2driv = gen2driv;
            this.ended = ended;
            t1 = new();
      endfunction

      task run();
            repeat(transaction_count)
            begin
                  t1.RandomGen();
                  //t1.display("Generator");
                  gen2driv.put(t1.copy());
                  #20;
            end
            ->ended;
      endtask
endclass
```

File Name	:	Driver.sv
Details	:	This file contains implementation of a class called driver.
Contents	:	

```
class driver;
      transaction t1;
      mailbox gen2driv;
      virtual intf vif;
      int transaction_count = 0;

      function new(virtual intf vif, mailbox gen2driv);
            this.vif = vif;
            this.gen2driv = gen2driv;
      endfunction

      task run();
            forever
                  begin
                        gen2driv.get(t1);
                        vif.Rst <= t1.Rst;
                        vif.Enqueue <= t1.Enqueue;
                        vif.Dequeue <= t1.Dequeue;
                        vif.Data_In <= t1.Data_In;
                        //t1.display("Driver");
                        @(posedge vif.clk);
                        transaction_count++;
            end
      endtask
endclass
```

File Name	:	Monitor.sv
Details	:	This file contains implementation of a class called monitor.
Contents	:	

```
class monitor;
      transaction t1;
      mailbox mon2scb;
      virtual intf vif;

      function new(virtual intf vif, mailbox mon2scb);
            this.vif = vif;
            this.mon2scb = mon2scb;
      endfunction

      task run();
            t1 = new();
            forever
            begin
                  @(posedge vif.clk);
                        #5;
                  t1.Rst = vif.Rst;
                        t1.Enqueue = vif.Enqueue;
                        t1.Dequeue = vif.Dequeue;
                        t1.Empty = vif.Empty;
                        t1.Full = vif.Full;
                        t1.Data_In = vif.Data_In;
                        t1.Data_Out = vif.Data_Out;

                        //t1.display("Monitor");
                  mon2scb.put(t1.copy());
            end
      endtask
endclass
```

File Name	:	Scoreboard.sv
Details	:	This file contains implementation of a class called scoreboard.
Contents	:	

```
class scoreboard;
      transaction t1;
      mailbox mon2scb;
    int transaction_count = 0;
  bit [15:0] queue[$:7];

      function new(mailbox mon2scb);
            int i = 0;
            this.mon2scb = mon2scb;
      endfunction
      task run();
           bit E, F;
           logic [15:0] dout;
           forever
           begin
          mon2scb.get(t1);
          E = 0;
          F = 0;
          dout = 16'bzzzzzzzzzzzzzzzz;
          if(t1.Rst == 1'b1)
          begin
            queue.delete();
            $display("Queue Reset");
            E = 1;
            F = 0;
            dout = 16'bzzzzzzzzzzzzzzzz;
          end
          else if(t1.Rst == 1'b0 && t1.Enqueue == 1'b1)
          begin
            if(queue.size() < 8)
            begin
              queue.push_front(t1.Data_In);
              dout = 16'bzzzzzzzzzzzzzzzz;
            end
          end
          else if(t1.Rst == 1'b0 && t1.Dequeue == 1'b1)
          begin
            if(queue.size() > 0)
              dout = queue.pop_back();
          end

          if(queue.size() == 0)
            E = 1;

          if(queue.size() == 8)
            F = 1;

          if(t1.Dequeue == 1'b1)
          begin
             if(t1.Data_Out === dout && t1.Empty == E && t1.Full ==
F)
            $display("Pass: Data_Out = %h, Empty = %b, Full = %b,
dout = %h, E = %b, F = %b",t1.Data_Out, t1.Empty, t1.Full, dout, E,
F);
           else
            $display("Fail");
           t1.Display("Scoreboard");
          end

          transaction_count++;
           end
      endtask
```

```
endclass
```

File Name	:	Environment.sv
Details	:	This file contains implementation of a class called environment which is a container for all the components of the testbench.

Contents :

```
`include "Transaction.sv"
`include "Generator.sv"
`include "Driver.sv"
`include "Monitor.sv"
`include "Scoreboard.sv"

class environment;
      generator gen;
      driver driv;
      monitor mon;
      scoreboard scb;
      mailbox m1, m2;
      virtual intf vif;
      event gen_ended;

      function new(virtual intf vif, int transaction_count);
            this.vif = vif;
            m1 = new();
            m2 = new();
            gen = new(m1, gen_ended);
            gen.transaction_count = transaction_count;

            driv = new(vif, m1);
            mon = new(vif, m2);
            scb = new(m2);
      endfunction

      task test();
      fork
            gen.run();
            driv.run();
            mon.run();
            scb.run();
         join_any
      endtask

      task run();
            test();
            wait(gen_ended.triggered);
            wait(gen.transaction_count == driv.transaction_count);
            wait(gen.transaction_count == scb.transaction_count);
            $finish;
      endtask
endclass
```

File Name	:	Test.sv
Details	:	This file contains implementation of a class called test which controls the simulation environment.
Contents	:	

```
`include "Environment.sv"

program test(intf i_intf);
      environment env;
      initial
      begin
           env = new(i_intf, 100);
            env.run();
      end
endprogram
```

File Name	:	Testbench.sv
Details	:	This file contains top module named tbench_top which connects interface and the instance of DUT.
Contents	:	

```
`include "Interface.sv"
`include "Test.sv"

module tbench_top;
      intf i_intf();
      test t1(i_intf);
      FIFO_Register F1
   (.clk(i_intf.clk), .Rst(i_intf.Rst), .Data_In(i_intf.Data_In),
.Enqueue(i_intf.Enqueue), .Data_Out(i_intf.Data_Out),
.Dequeue(i_intf.Dequeue), .Empty(i_intf.Empty), .Full(i_intf.Full));

      initial
      begin
            i_intf.clk <= 0;
      end

      always #10 i_intf.clk <= ~i_intf.clk;
endmodule
```

4.3.4. Introduced Errors

Sl.No	Encrypted source code file	Errors	Source code file
2.	RAM_256B_i1.vp	No Errors	RAM_256B_i1.v

5. Demonstration of System Verilog capabilities

5.1. Demonstration of usage of randcase

‘randcase’ randomly selects one of the branches randomly. The case item expressions are positive integers representing weights associated to the item. The probability of selecting an item can be obtained as weight of item/sum of all weights.

Syntax:

```
randcase
Item1 : Statements<1>;
Item2 : Statements<2>;
…
endcase
```

Example:

```
module test;
  int totalCount = 0;
  int count1 = 0, count2 = 0, count3 = 0;
  initial begin
    repeat(10)
    begin
      randcase
      1: begin
            count1++;
            $display("First branch");
         end
      2: begin
            count2++;
            $display("Second branch");
         end
      3: begin
            count3++;
            $display("Third branch");
         end
      endcase
      totalCount++;
    end
    $display("Visited: First branch %0d, Second branch %d, Third branch 
%0d",count1, count2, count3);
    $display("Probability: First branch %f, Second branch %f, Third branch 
%f",count1*1.0/totalCount, count2*1.0/totalCount, count3*1.0/totalCount);
  end
endmodule
```

In this example, the total weight is 1+2+3 = 6. Thus probability of first branch being selected is 1/6, second branch being selected is 2/6 and third branch being selected is 3/6.
NOTE: Increase the repeat count from 10 to 100 and then to 1000 to observe the difference.

5.2. Generation of 32 bit random numbers using system functions

System Function	Description
$urandom(int seed)	Returns 32 bit unsigned random number
$random( )	Returns 32 bit signed random number
$urandom_range(int low, int high)	Returns 32 bit unsigned random number bounded by low and high

Example:

```
module test;
  bit [31:0] data1;
  bit [31:0] data2;
  bit [31:0] data3;
  initial
  begin
    repeat(10)
    begin
      data1 = $urandom();
      data2 = $random();
      data3 = $urandom_range(10,25000);
      $display("data1 = %0d, data2 = %0d, data3 = %0d", data1, data2, data3);
    end
  end
endmodule
```

5.3. Using enumerated types

Enumerated type defines a set of named values. The simplest enumerated type declaration contains a list of constant names and one or more variables.

Method	Description
first()	returns the value of the first member of the enumeration
last()	returns the value of the last member of the enumeration
next()	returns the value of next member of the enumeration
next(N)	returns the value of next Nth member of the enumeration
prev()	returns the value of previous member of the enumeration
prev(N)	returns the value of previous Nth member of the enumeration
num()	returns the number of elements in the given enumeration
name()	returns the string representation of the given enumeration value

Example 1:

```
module test;

  enum { Sunday, Monday, Tuesday, Wednesday = 5, Thursday, Firday, Saturday}
Days_Of_Week;

  int i = 0;

  initial begin
      Days_Of_Week = Days_Of_Week.first();
    for(i = 0; i < 7; i++)
    begin
      $display("Days_Of_Week = %s, Value = %0d",Days_Of_Week.name(),
Days_Of_Week);
      Days_Of_Week = Days_Of_Week.next();
    end
  end
endmodule
```

Example 2:

```
module test;

  typedef enum { Sunday, Monday, Tuesday, Wednesday = 5, Thursday, Firday,
Saturday} Days_Of_Week;

  Days_Of_Week today;

  initial begin
    today = Wednesday;
    $display("today is %s, value is %0d",today.name(), today);
    today++;
    $display("today is %s, value is %0d",today.name(), today);
    today = today.first();
    if(today == Sunday)
      $display("Holiday");
    else
      $display("Working day");
  end
endmodule
```

Sample Viva Voce Questions

1. How is system Verilog different than Verilog?
2. What is the difference between programming language and description language?
3. What are the different datatypes available in system Verilog?
4. What is the syntax of foreach loop?
5. Differentiate between system Verilog task and function.
6. Define class
7. What is a constructor of the class?
8. Differentiate between stack and queue datastructures.
9. What is a structure?
10. Define enumeration
11. Why is typedef key word used?
12. What is an interface?
13. Difference between $random() and $urandom_int() functions?
14. What the different format specifiers that can be used with $display() function?
15. What is $rand_case?
16. What is a layered testbench architechture?
17. What are mailboxes?
18. What is a generator?
19. What is a transaction?
20. What is a driver?
21. What are monitor and scoreboard?
22. What are constrains and how are they defined in the code?
23. What are implication constraints?
24. Define test environment
25. Differentiate rand and randc modifiers