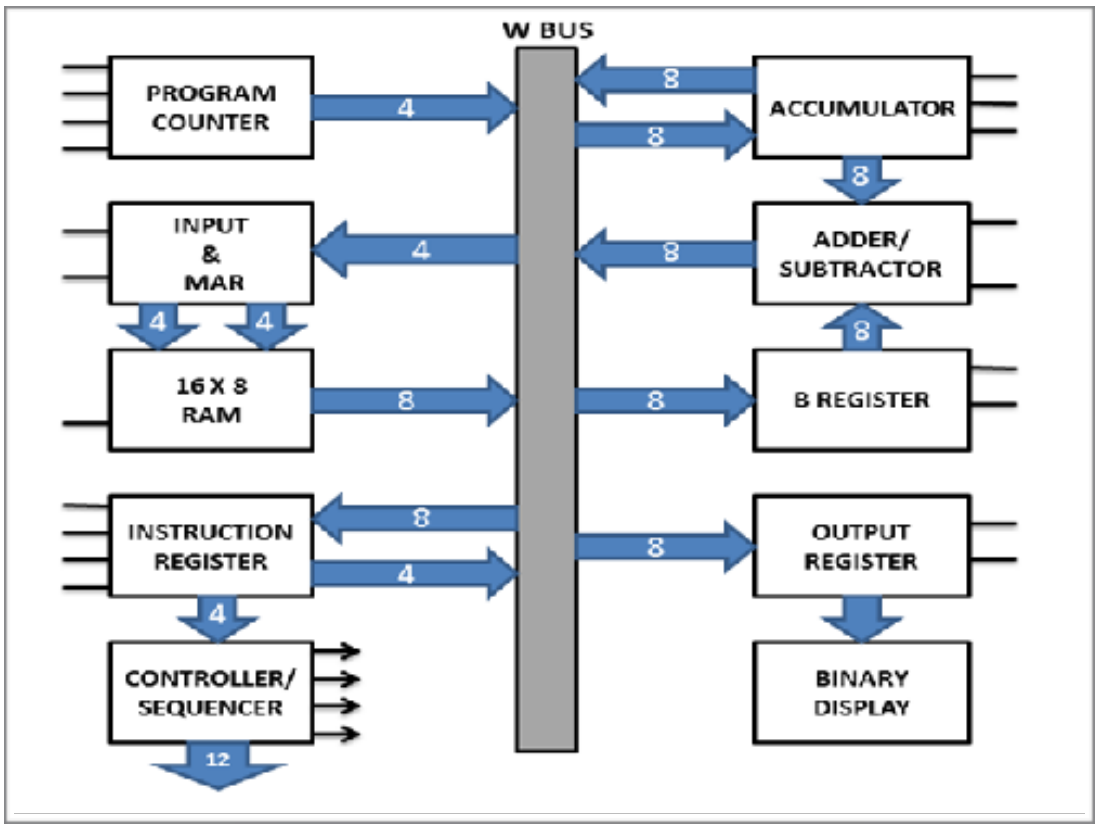
Assignment-6

SAP-1 Report



Aditya Shridhar Hegde

IMT2016054

Screenshots

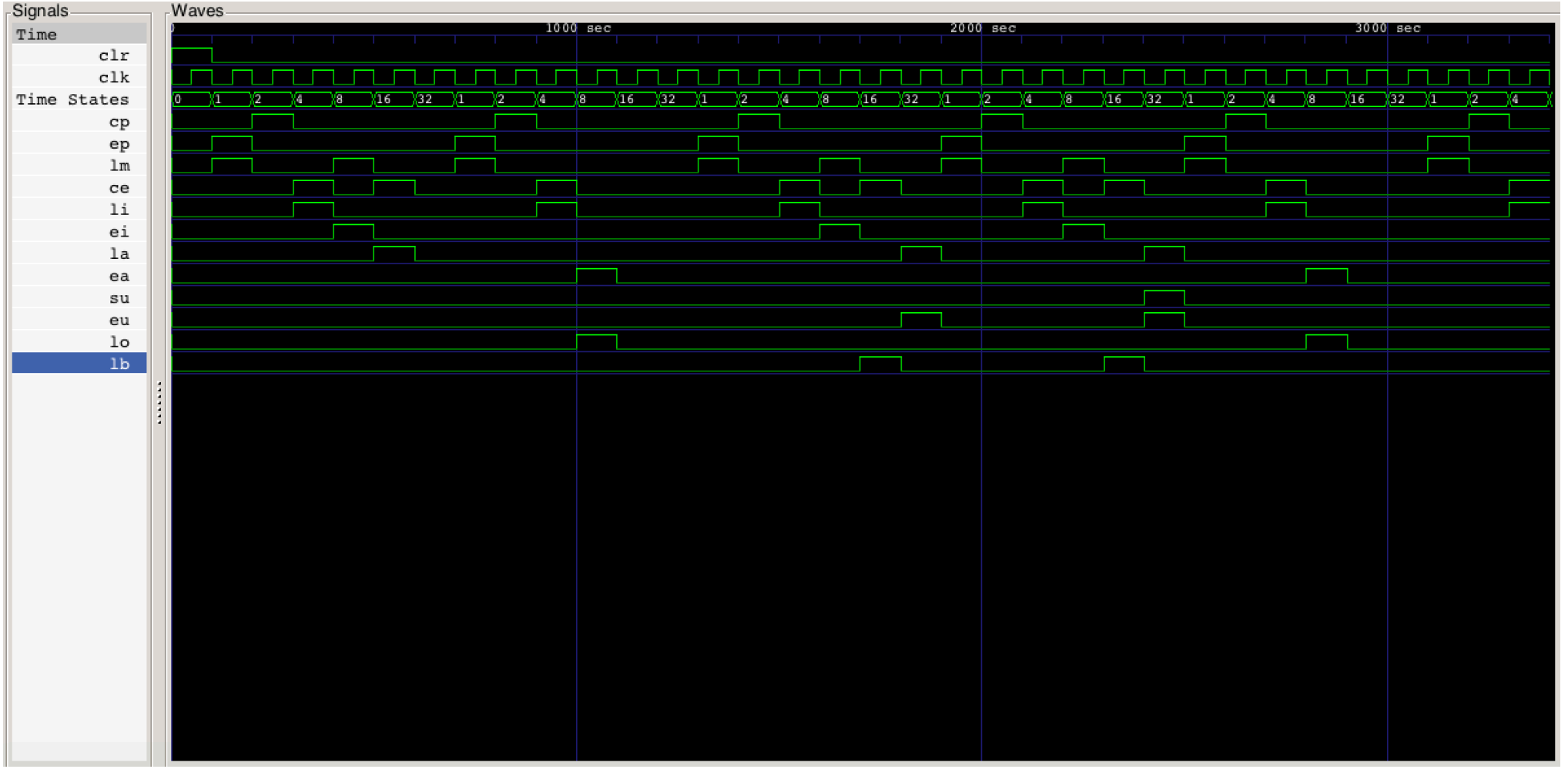
1. **Control Signals**

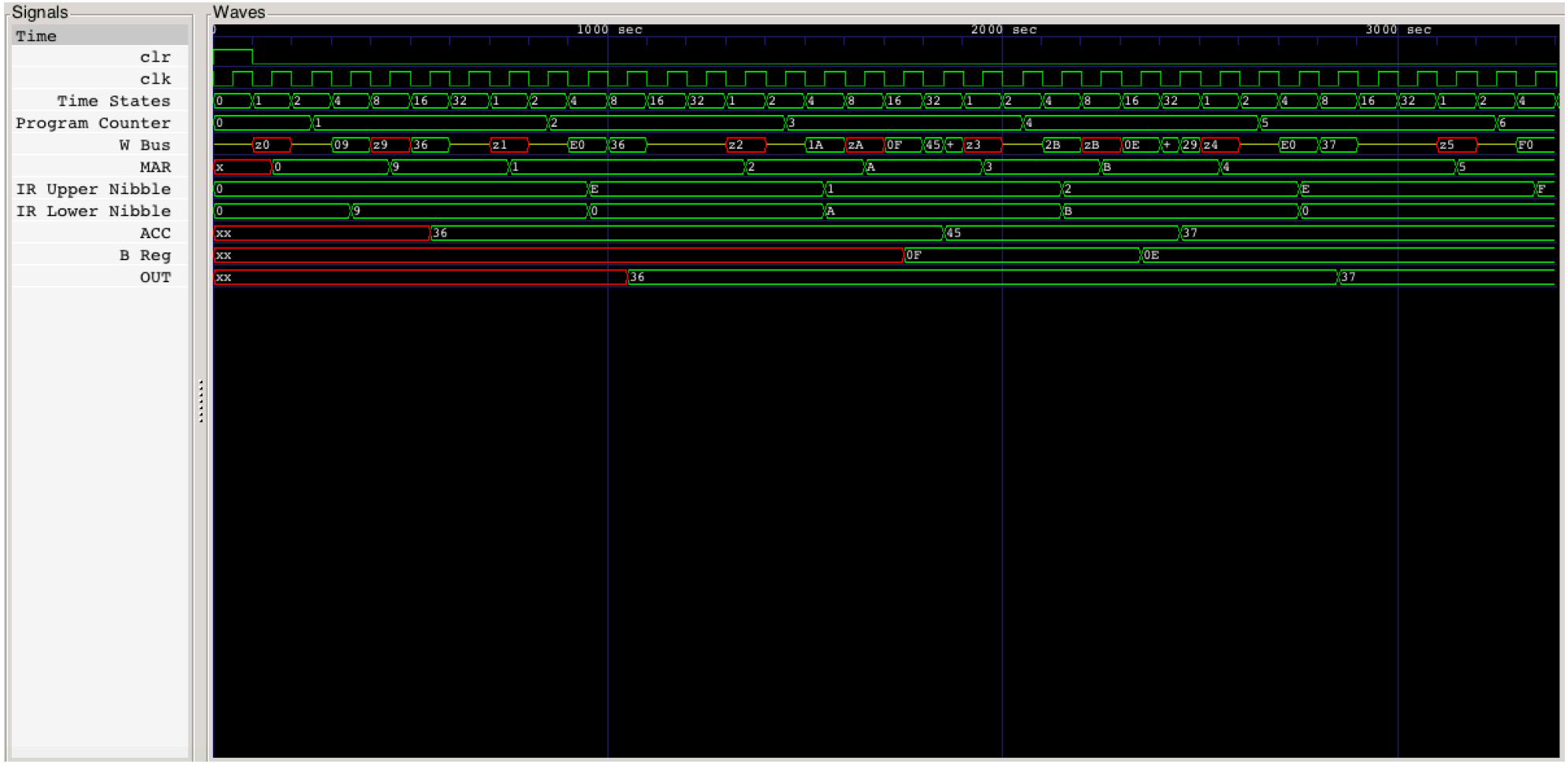
   Page 3 contains the control signals plotted against time. The value of the time states are displayed in decimal.

2. **Registers**

   Page 4 contains the values in the main registers of the CPU plotted against time. The values of Time States, Program Counter, Accumulator (AC), B register (B Reg) and Output register (OUT) are displayed in decimal while the values of MAR, W Bus and Instruction Register (IR) are displayed in hexadecimal.

The waveforms have been generated using the gtkwave tool.


Signals
Waves
Time
clr
clk
Time States
cp
ep
lm
ce
li
ei
la
ea
su
eu
lo
lb
1000 sec
2000 sec
3000 sec
0 1 2 4 8 16 32 1 2 4 8 16 32 1 2 4 8 16 32 1 2 4 8 16 32 1 2 4 8 16 32 1 2 4


Signals
Waves
Time
clr
clk
Time States
Program Counter
W Bus
MAR
IR Upper Nibble
IR Lower Nibble
ACC
B Reg
OUT
1000 sec
2000 sec
3000 sec

Code

1. **Tri State Buffer**

```
module TRI_STATE
    #(parameter w = 8)
    (input [w-1 : 0] i, input en, output [w-1 : 0] o);

    assign o = en ? i : 'bz;
endmodule
```

2. **Ring Counter**

```
module RC_SAP
    (
        input clk, clr,
        output reg [5 : 0] T
    );

    always @ (negedge clk, posedge clr) begin
        if(clr) T = 6'b000000;
        else if(T == 6'b000000) T[0] = 1'b1;
        else if(T == 6'b100000) T = 6'b000001;
        else T = T << 1;
    end
endmodule
```

3. **Control Sequencer**

```
module CONTROL_SAP
    (
        input [3 : 0] opcode,
        input clk, clr,
        output reg [11 : 0] cntrlcodes
    );
```

```
wire [5 : 0] tstate;
RC_SAP counter (clk, clr, tstate);

always @ (negedge clk) begin
        casez ({tstate, opcode})
                {6'b000000, 4'b????} : cntrlcodes <= 12'h000;
                //T0 state
                {6'b000001, 4'b????} : cntrlcodes <= 12'h600;
                //T1 state
                {6'b000010, 4'b????} : cntrlcodes <= 12'h800;
                //T2 state
                {6'b000100, 4'b????} : cntrlcodes <= 12'h180;

                //LDA T3 state
                {6'b001000, 4'b0000} : cntrlcodes <= 12'h240;
                //LDA T4 state
                {6'b010000, 4'b0000} : cntrlcodes <= 12'h120;
                //LDA T5 state
                {6'b100000, 4'b0000} : cntrlcodes <= 12'h000;

                //ADD T3 state
                {6'b001000, 4'b0001} : cntrlcodes <= 12'h240;
                //ADD T4 state
                {6'b010000, 4'b0001} : cntrlcodes <= 12'h102;
                //ADD T5 state
                {6'b100000, 4'b0001} : cntrlcodes <= 12'h024;

                //SUB T3 state
                {6'b001000, 4'b0010} : cntrlcodes <= 12'h240;
                //SUB T4 state
                {6'b010000, 4'b0010} : cntrlcodes <= 12'h102;
                //SUB T5 state
                {6'b100000, 4'b0010} : cntrlcodes <= 12'h02C;
```

```
                    //OUT T3 state
                    {6'b001000, 4'b1110} : cntrlcodes <= 12'h011;
                    //SUB T4 state
                    {6'b010000, 4'b1110} : cntrlcodes <= 12'h000;
                    //SUB T5 state
                    {6'b100000, 4'b1110} : cntrlcodes <= 12'h000;

                    //HLT T3 state
                    {6'b001000, 4'b1111} : $finish;
                endcase
        end
    endmodule
```

4. **Program Counter**

```
    module PC_SAP
        (
            input clk, clr, cp, ep,
            output [3 : 0] toBUS
        );

        reg [3 : 0] counter;
        TRI_STATE #(4) bus (counter, ep, toBUS);

        always @ (posedge clk, posedge clr)
            if(clr) counter <= 4'h0;
            else if(cp) counter <= counter + 1;
    endmodule
```

5. **MAR**

```
    module MAR_SAP
        (
            input clk, lm,
            input [3 : 0] fromBus,
```

```
        output reg [3 : 0] toRAM
    );

    always @ (posedge clk)
        if(lm) toRAM <= fromBus;
endmodule
```

6. **RAM**

```
module RAM_SAP
    (
        input ce,
        input [3 : 0] fromMAR,
        output [7 : 0] toBus
    );

    wire [7 : 0] mem[15 : 0];

    assign mem[0] = 8'h19;
    assign mem[9] = 8'hFF;

    TRI_STATE bus (mem[fromMAR], ce, toBus);
endmodule
```

7. **Instruction Register**

```
module IR_SAP
    (
        input clk, clr, li, ei,
        input [7 : 0] in,
        output [3 : 0] tobus,
        output reg [3 : 0] tocs
    );

    reg [3 : 0] temp;
```

```
        TRI_STATE #(4) bus (temp, ei, tobus);

        always @ (posedge clk, posedge clr)
            if(clr) {tocs, temp} <= 8'h00;
            else if(li) {tocs, temp} <= in;
    endmodule
```

8. **Accumulator**

```
    module ACC_SAP
        (
            input clk, la, ea,
            input [7 : 0] fromBus,
            output [7 : 0] toBus,
            output reg [7 : 0] toALU
        );

        TRI_STATE bus (toALU, ea, toBus);

        always @ (posedge clk)
            if(la) toALU <= fromBus;
    endmodule
```

9. **Addition/Subtraction**

```
    module ALU_SAP
        (
            input su, eu,
            input [7 : 0] fromAcc, fromBreg,
            output [7 : 0] toBus
        );

        wire [7 : 0] ans;
        TRI_STATE bus (ans, eu, toBus);

        assign ans = su ? (fromAcc - fromBreg) : (fromAcc + fromBreg);
```

```
endmodule
```

10. **B Register**

```
module B_SAP
    (
        input clk, lb,
        input [7 : 0] fromBus,
        output reg [7 : 0] toALU
    );

    always @ (posedge clk)
        if(lb) toALU <= fromBus;
endmodule
```

11. **Output Register**

```
module OUT_SAP
    (
        input clk, lo,
        input [7 : 0] fromBus,
        output reg [7 : 0] out
    );

    always @ (posedge clk)
        if(lo) out <= fromBus;
endmodule
```

12. **SAP-1**

```
module SAP (input clk, clr);
    wire [7 : 0] bus, fromACC, fromBREG, out;
    wire cp, ep, lm, ce, li, ei, la, ea, su, eu, lb, lo;
    wire [3 : 0] fromIR, fromMAR;

    CONTROL_SAP cs (fromIR, clk, clr, {cp, ep, lm, ce, li, ei, la, ea, su, eu, lb,
lo});
```

```
        IR_SAP ir (clk, clr, li, ei, bus, bus[3 : 0], fromIR);
        MAR_SAP mar (clk, lm, bus[3 : 0], fromMAR);
        RAM_SAP ram (ce, fromMAR, bus);
        PC_SAP pc (clk, clr, cp, ep, bus[3 : 0]);
        ACC_SAP acc (clk, la, ea, bus, bus, fromACC);
        ALU_SAP alu (su, eu, fromACC, fromBREG, bus);
        B_SAP breg (clk, lb, bus, fromBREG);
        OUT_SAP outreg (clk, lo, bus, out);

        //Display Module Equivalent
        always @ (negedge clk)
                if(lo) $display("OUTPUT (Decimal): %d", out);
endmodule
```

13. **Test Bench for Ring Counter**

```
`include "RC_SAP.v"

module TB_RC_SAP;
        wire [5 : 0] tstate;
        reg clk, clr;

        RC_SAP counter (clk, clr, tstate);

        initial begin
                clr = 1;
                clk = 0;
                #25 clr = 0;
                #1250 $finish;
        end

        always #50 clk = ~clk;

        always @ (posedge clk) begin
                $display("State: %b", tstate);
```

```
        end
endmodule
```

Output of Test Bench:

```
State: 000000
State: 000001
State: 000010
State: 000100
State: 001000
State: 010000
State: 100000
State: 000001
State: 000010
State: 000100
State: 001000
State: 010000
State: 100000
```

14. **Test Bench for SAP-1**

```
`include "TRI_STATE.v"
`include "RC_SAP.v"
`include "CONTROL_SAP.v"
`include "IR_SAP.v"
`include "RAM_SAP.v"
`include "MAR_SAP.v"
`include "PC_SAP.v"
`include "ACC_SAP.v"
`include "B_SAP.v"
`include "ALU_SAP.v"
`include "OUT_SAP.v"
`include "SAP.v"

module TB_SAP;
        reg clk, clr;
```

```
    SAP sap (clk, clr);

    initial begin
        // $dumpfile("SAP.vcd");
        // $dumpvars(0, TB_SAP);
        clr = 1;
        clk = 0;

        //Program RAM
        force sap.ram.mem[0] = 8'h09;
        force sap.ram.mem[1] = 8'hE0;
        force sap.ram.mem[2] = 8'h1A;
        force sap.ram.mem[3] = 8'h2B;
        force sap.ram.mem[4] = 8'hE0;
        force sap.ram.mem[5] = 8'hF0;
        force sap.ram.mem[9] = 8'h36;
        force sap.ram.mem[10] = 8'h0F;
        force sap.ram.mem[11] = 8'h0E;

        #100 clr = 0;
    end

    always #50 clk = ~clk;
endmodule
```

Output of Test Bench:

```
    OUTPUT (Decimal):  54
    OUTPUT (Decimal):  55
```