Digital Design and Computer Organization Laboratory

UE20CS206

3rd Semester, Academic Year 2020-21

Date:

Name : Sriram R	SRN : PES1UG20CS435	Section : H

Experiment Number: 1 Week # : 6

Title of the Program: Program counter simulation on iverilog

Code:

pc.v :

```
// library modules

module invert (input wire i, output wire o);
   assign o = !i;
endmodule
```

```
module and2 (input wire i0, i1, output wire o);
  assign o = i0 & i1;
endmodule

module or2 (input wire i0, i1, output wire o);
  assign o = i0 | i1;
endmodule

module xor2 (input wire i0, i1, output wire o);
  assign o = i0 ^ i1;
endmodule

module nand2 (input wire i0, i1, output wire o);
   wire t;
   and2 and2_0 (i0, i1, t);
   invert invert_0 (t, o);
endmodule

module nor2 (input wire i0, i1, output wire o);
   wire t;
   or2 or2_0 (i0, i1, t);
   invert invert_0 (t, o);
```

```
endmodule


module xnor2 (input wire i0, i1, output wire o);
  wire t;
  xor2 xor2_0 (i0, i1, t);
  invert invert_0 (t, o);
endmodule


module and3 (input wire i0, i1, i2, output wire o);
  wire t;
  and2 and2_0 (i0, i1, t);
  and2 and2_1 (i2, t, o);
endmodule


module or3 (input wire i0, i1, i2, output wire o);
  wire t;
  or2 or2_0 (i0, i1, t);
  or2 or2_1 (i2, t, o);
endmodule


module nor3 (input wire i0, i1, i2, output wire o);
  wire t;
  or2 or2_0 (i0, i1, t);
  nor2 nor2_0 (i2, t, o);
endmodule

module nand3 (input wire i0, i1, i2, output wire o);
  wire t;
  and2 and2_0 (i0, i1, t);
  nand2 nand2_1 (i2, t, o);
endmodule

module xor3 (input wire i0, i1, i2, output wire o);
  wire t;
  xor2 xor2_0 (i0, i1, t);
  xor2 xor2_1 (i2, t, o);
endmodule

module xnor3 (input wire i0, i1, i2, output wire o);
  wire t;
  xor2 xor2_0 (i0, i1, t);
  xnor2 xnor2_0 (i2, t, o);
endmodule
```

```
module mux2 (input wire i0, i1, j, output wire o);
  assign o = (j==0)?i0:i1;
endmodule

module mux4 (input wire [0:3] i, input wire j1, j0, output wire o);
  wire  t0, t1;
  mux2 mux2_0 (i[0], i[1], j1, t0);
  mux2 mux2_1 (i[2], i[3], j1, t1);
  mux2 mux2_2 (t0, t1, j0, o);
endmodule

module mux8 (input wire [0:7] i, input wire j2, j1, j0, output wire o);
  wire  t0, t1;
  mux4 mux4_0 (i[0:3], j2, j1, t0);
  mux4 mux4_1 (i[4:7], j2, j1, t1);
  mux2 mux2_0 (t0, t1, j0, o);
endmodule

module demux2 (input wire i, j, output wire o0, o1);
  assign o0 = (j==0)?i:1'b0;
  assign o1 = (j==1)?i:1'b0;
endmodule
```

```
module demux4 (input wire i, j1, j0, output wire [0:3] o);
  wire  t0, t1;
  demux2 demux2_0 (i, j1, t0, t1);
  demux2 demux2_1 (t0, j0, o[0], o[1]);
  demux2 demux2_2 (t1, j0, o[2], o[3]);
endmodule

module demux8 (input wire i, j2, j1, j0, output wire [0:7] o);
  wire  t0, t1;
  demux2 demux2_0 (i, j2, t0, t1);
  demux4 demux4_0 (t0, j1, j0, o[0:3]);
  demux4 demux4_1 (t1, j1, j0, o[4:7]);
endmodule

module df (input wire clk, in, output wire out);
  reg df_out;
  always@(posedge clk) df_out <= in;
  assign out = df_out;
endmodule

module dfr (input wire clk, reset, in, output wire out);
```

```
  wire reset_, df_in;
  invert invert_0 (reset, reset_);
  and2 and2_0 (in, reset_, df_in);
  df df_0 (clk, df_in, out);
endmodule

module dfrl (input wire clk, reset, load, in, output wire out);
  wire _in;
  mux2 mux2_0(out, in, load, _in);
  dfr dfr_1(clk, reset, _in, out);
endmodule

module fulladd(input wire a, b, cin, output wire sum, cout);

    wire [4:0] t;
    xor2 x0(a, b, t[0]);
    xor2 x1(t[0], cin, sum);

    and2 a0(a, b, t[1]);
    and2 a1(a, cin, t[2]);
    and2 a2(b, cin, t[3]);
```

```
   or2 o0(t[1], t[2], t[4]);
   or2 o1(t[3], t[4], cout);

endmodule

// Write code for modules you need here

module pc_slice (input wire offset, inc, sub, cin, load, clk, reset,
output wire cout, pc);

// Declare wires here

       wire inc1;
       wire andout;
       wire xorout;
       wire sum;

// Instantiate modules here

       invert n0(inc, inc1);
       and2 a0(offset, inc1, andout);
       xor2 x0(andout, sub, xorout);
```

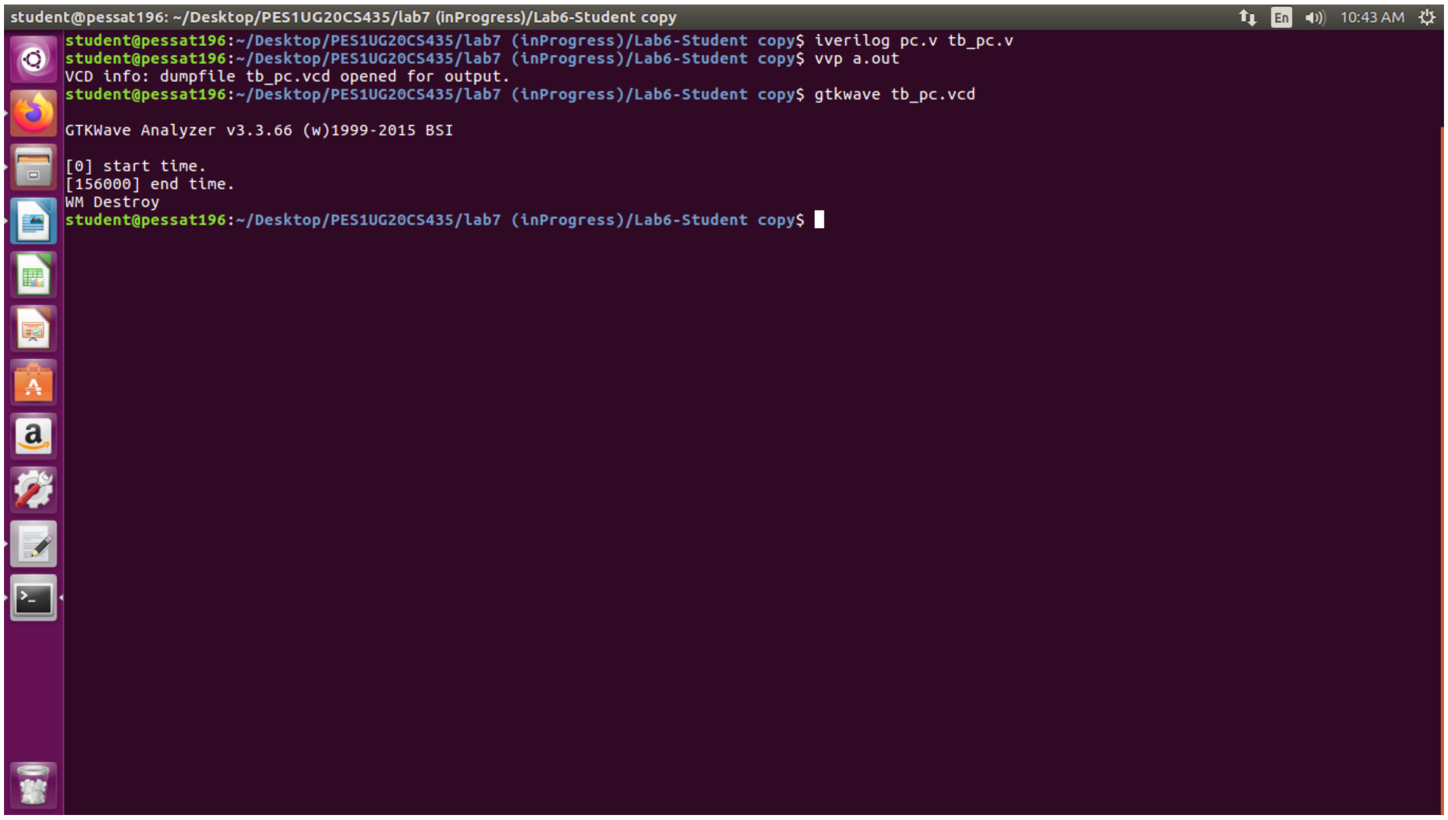
```
	fulladd f0(pc, xorout, cin, sum, cout);

	dfrl d0(clk, reset, load, sum, pc);

endmodule

module pc (input wire clk, reset, inc, add, sub, input wire [15:0]
offset, output wire [15:0] pc);

// Declare wires here
	wire [0:15] c;

// Instantiate modules here

	pc_slice p0(offset[0], inc, sub, sub, load, clk, reset, c[0], pc[0]);
	pc_slice p1(offset[1], inc, sub, c[0], load, clk, reset, c[1], pc[1]);
	pc_slice p2(offset[2], inc, sub, c[1], load, clk, reset, c[2], pc[2]);
	pc_slice p3(offset[3], inc, sub, c[2], load, clk, reset, c[3], pc[3]);
	pc_slice p4(offset[4], inc, sub, c[3], load, clk, reset, c[4], pc[4]);
	pc_slice p5(offset[5], inc, sub, c[4], load, clk, reset, c[5], pc[5]);
	pc_slice p6(offset[6], inc, sub, c[5], load, clk, reset, c[6], pc[6]);
	pc_slice p7(offset[7], inc, sub, c[6], load, clk, reset, c[7], pc[7]);
	pc_slice p8(offset[8], inc, sub, c[7], load, clk, reset, c[8], pc[8]);
	pc_slice p9(offset[9], inc, sub, c[8], load, clk, reset, c[9], pc[9]);
	pc_slice p10(offset[10], inc, sub, c[9], load, clk, reset, c[10],
pc[10]);
	pc_slice p11(offset[11], inc, sub, c[10], load, clk, reset, c[11],
pc[11]);
	pc_slice p12(offset[12], inc, sub, c[11], load, clk, reset, c[12],
pc[12]);
	pc_slice p13(offset[13], inc, sub, c[12], load, clk, reset, c[13],
pc[13]);
	pc_slice p14(offset[14], inc, sub, c[13], load, clk, reset, c[14],
pc[14]);
	pc_slice p15(offset[15], inc, sub, c[14], load, clk, reset, c[15],
pc[15]);

endmodule
```

tb_pc.v :

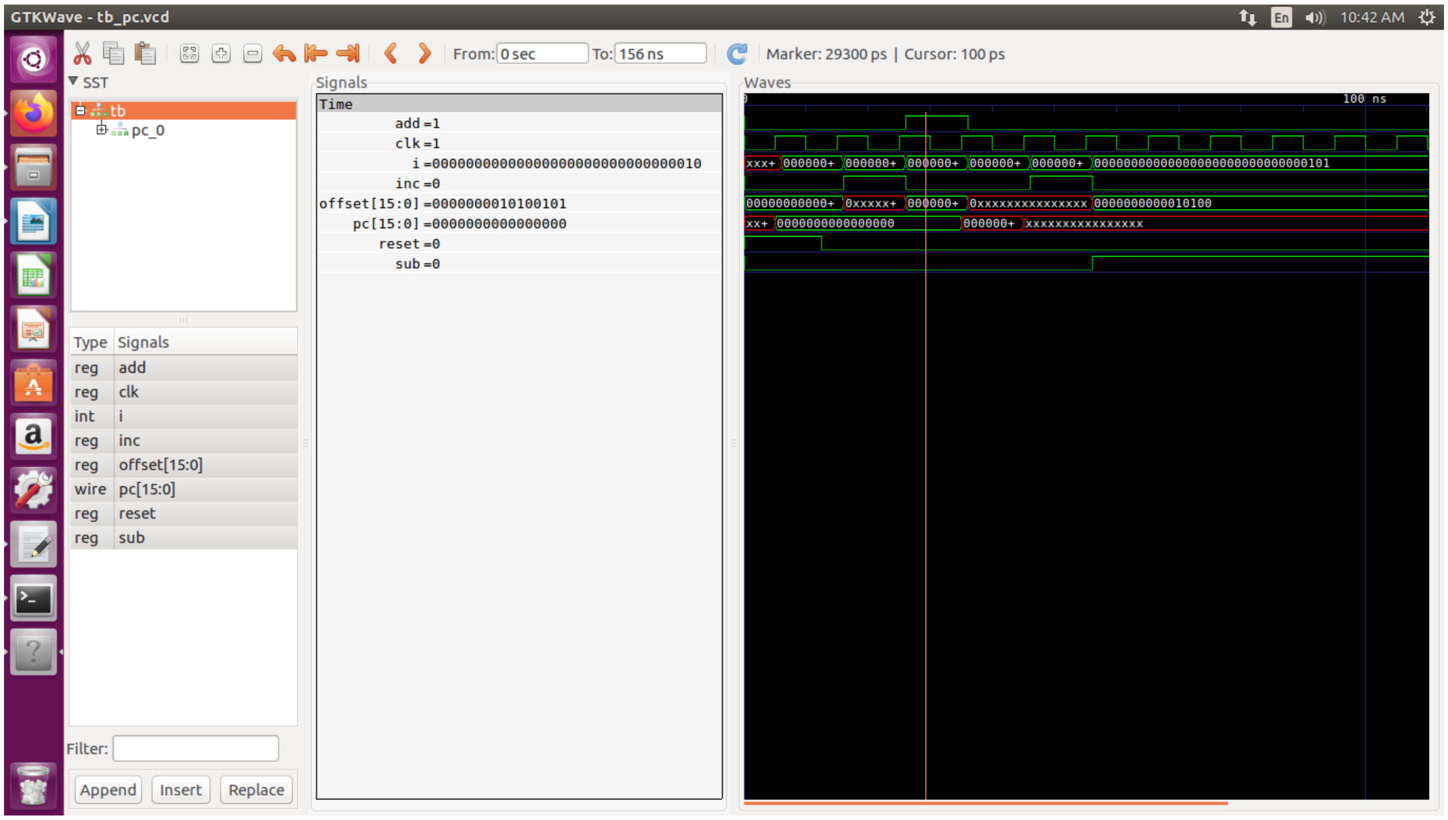
```
`timescale 1 ns / 100 ps
`define TESTVECS 5
```

```
module tb;
  reg clk, reset, inc, add, sub;
  reg [15:0] offset;
  wire [15:0] pc;
  reg [18:0] test_vecs [0:(`TESTVECS-1)];
  integer i;
  initial begin $dumpfile("tb_pc.vcd"); $dumpvars(0,tb); end
  initial begin reset = 1'b1; #12.5 reset = 1'b0; end
  initial clk = 1'b0; always #5 clk =~ clk;
  initial begin
    test_vecs[0][18] = 1'b1; test_vecs[0][17] = 1'b0; test_vecs[0][16] =
1'b0;
    test_vecs[0][15:0] = 15'hxx;
    test_vecs[1][18] = 1'b0; test_vecs[1][17] = 1'b1; test_vecs[1][16] =
1'b0;
    test_vecs[1][15:0] = 15'ha5;
    test_vecs[2][18] = 1'b0; test_vecs[2][17] = 1'b0; test_vecs[2][16] =
1'b0;
    test_vecs[2][15:0] = 15'hxx;
    test_vecs[3][18] = 1'b1; test_vecs[3][17] = 1'b0; test_vecs[3][16] =
1'b0;
    test_vecs[3][15:0] = 15'hxx;
    test_vecs[4][18] = 1'b0; test_vecs[4][17] = 1'b0; test_vecs[4][16] =
1'b1;
    test_vecs[4][15:0] = 15'h14;
  end
  initial {inc, add, sub, offset} = 0;
  pc pc_0 (clk, reset, inc, add, sub, offset, pc);
  initial begin
    #6 for(i=0;i<`TESTVECS;i=i+1)
      begin #10 {inc, add, sub, offset}=test_vecs[i]; end
    #100 $finish;
  end
endmodule
```

Output waveform



```
student@pessat196: ~/Desktop/PES1UG20CS435/lab7 (inProgress)/Lab6-Student copy
student@pessat196:~/Desktop/PES1UG20CS435/lab7 (inProgress)/Lab6-Student copy$ iverilog pc.v tb_pc.v
student@pessat196:~/Desktop/PES1UG20CS435/lab7 (inProgress)/Lab6-Student copy$ vvp a.out
VCD info: dumpfile tb_pc.vcd opened for output.
student@pessat196:~/Desktop/PES1UG20CS435/lab7 (inProgress)/Lab6-Student copy$ gtkwave tb_pc.vcd

GTKWave Analyzer v3.3.66 (w)1999-2015 BSI

[0] start time.
[156000] end time.
WM Destroy
student@pessat196:~/Desktop/PES1UG20CS435/lab7 (inProgress)/Lab6-Student copy$
```



GTKWave - tb_pc.vcd

From: 0 sec To: 156 ns Marker: 29300 ps | Cursor: 100 ps

SST: tb, pc_0

Type	Signals
reg	add
reg	clk
int	i
reg	inc
reg	offset[15:0]
wire	pc[15:0]
reg	reset
reg	sub

```
Time
          add =1
          clk =1
            i =00000000000000000000000000000010
          inc =0
offset[15:0] =0000000010100101
    pc[15:0] =0000000000000000
        reset =0
          sub =0
```

```
i:      xxx+ 000000+ 000000+ 000000+ 000000+ 000000+ 00000000000000000000000000000101
offset: 00000000000+ 0xxxxx+ 000000+ 0xxxxxxxxxxxxxx 0000000000010100
pc:     xx+ 0000000000000000 000000+ xxxxxxxxxxxxxxxx
```