Vishal S Salanke
PES1UG21CS843

Digital Design & Computer Organization
Lab Report

Experiment 1:- Basic Gates

Iverilog Code:-

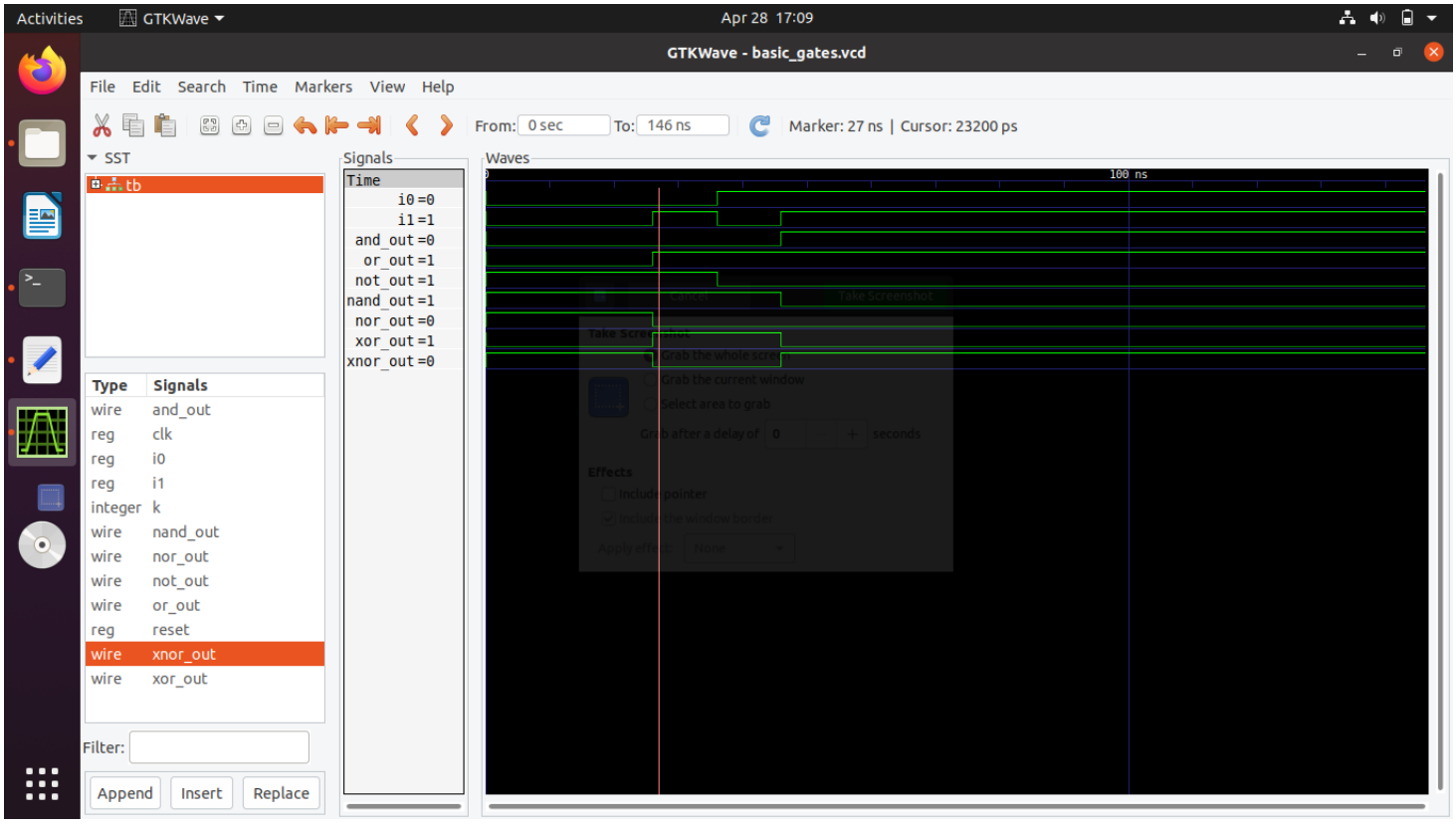
```
module basic_gates(input wire i0, i1, output and_out, or_out, not_out, xor_out,
        nand_out, nor_out, xnor_out);
                and2 and2_0(i0, i1, and_out);
                or2 or2_0(i0, i1, or_out);
                invert invert_0(i0, not_out);
                xor2 xor2_0(i0, i1, xor_out);
                nand2 nand_0(i0, i1, nand_out);
                nor2 nor_0(i0, i1, nor_out);
                xnor2 xnor2_0(i0, i1, xnor_out);
endmodule
```

Test Bench:-

```
`timescale 1 ns / 100 ps
`define TESTVECS 4
module tb;
reg clk, reset;
reg i0, i1;
wire and_out, or_out, not_out, xor_out, nand_out, nor_out, xnor_out;
reg [1:0] test_vecs [0:(`TESTVECS-1)];
integer k;
initial begin $dumpfile("basic_gates.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] = 6'b00;
test_vecs[1] = 6'b01;
test_vecs[2] = 6'b10;
test_vecs[3] = 6'b11;
```

```
end
initial {i0, i1} = 0;
basic_gates basic_gates_0(i0, i1,and_out, or_out, not_out, xor_out, nand_out,
nor_out, xnor_out);
initial begin
#6 for(k=0;k<`TESTVECS;k=k+1)
begin #10 {i0, i1}=test_vecs[k];
end
#100 $finish;
end
endmodule
```

OUTPUT



Experiment 2:- Half Adder and Full Adder

Iverilog Code:-

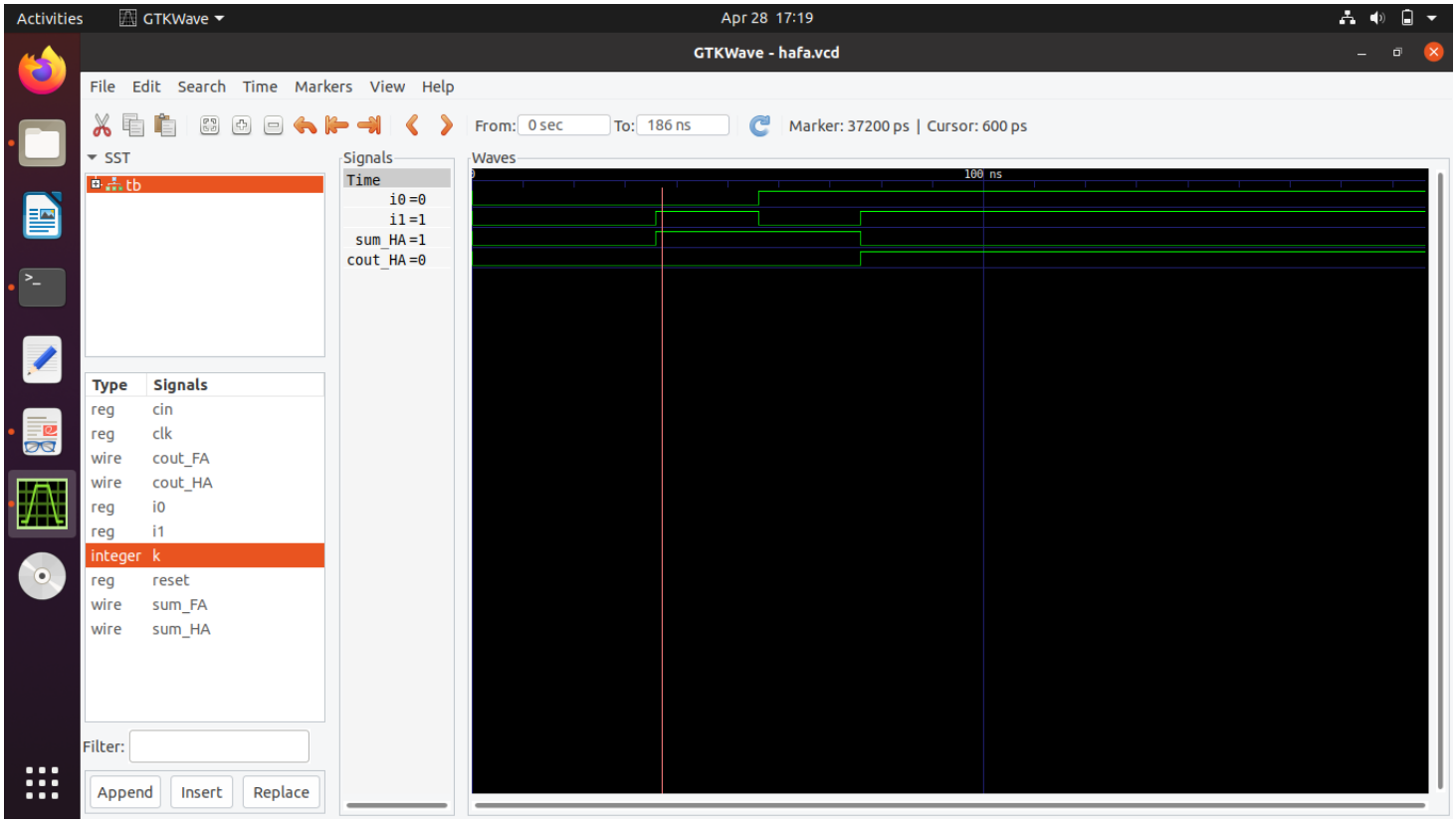
```
module halfAdder(input a, b, output wire sum, carry);
	xor2 xor2_0(a, b, sum);
	and2 and2_0(a, b, carry);
endmodule

module fullAdder(input wire a, b, cin, output wire sum, cout);
	wire [0:4] t;
	xor2 xor2_1(a, b, t[0]);
	xor2 xor2_2(t[0], cin, sum);
	and2 and2_1(a, b, t[1]);
	and2 and2_2(a, cin, t[2]);
	and2 and2_3(b, cin, t[3]);
	or2 or2_0(t[1], t[2], t[4]);
	or2 or2_1(t[4], t[3], cout);
endmodule
```

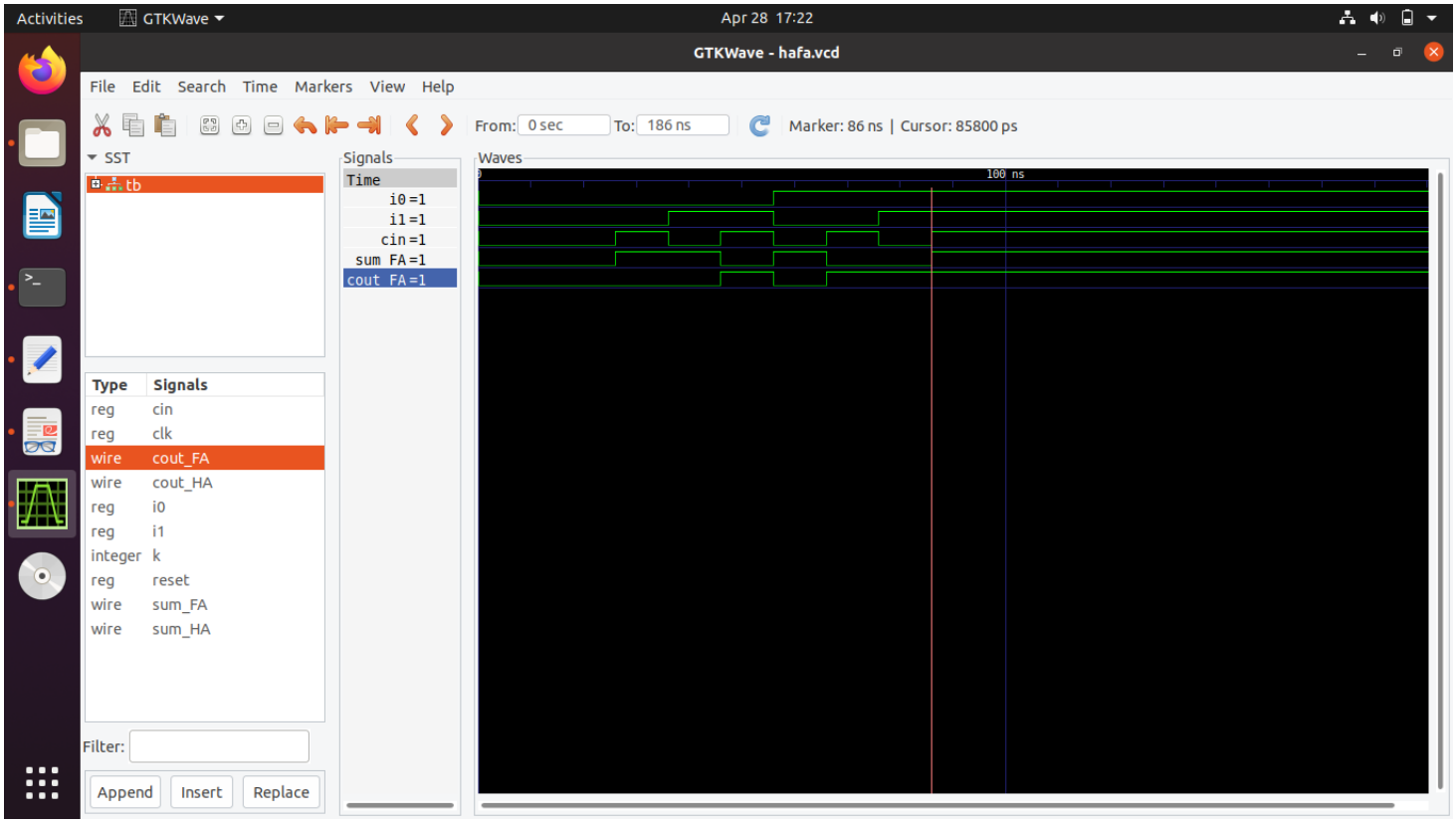
Test Bench:-

```
`timescale 1 ns / 100 ps
`define TESTVECS 8
module tb;
reg clk, reset;
reg i0, i1, cin;
wire sum_HA, cout_HA,sum_FA, cout_FA;
reg [2:0] test_vecs [0:(`TESTVECS-1)];
integer k;
initial begin $dumpfile("hafa.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] = 3'b000;
test_vecs[1] = 3'b001;
```

```
test_vecs[2] = 3'b010;
test_vecs[3] = 3'b011;
test_vecs[4] = 3'b100;
test_vecs[5] = 3'b101;
test_vecs[6] = 3'b110;
test_vecs[7] = 3'b111;
end
initial {i0, i1, cin} = 0;
halfAdder halfAdder_0(i0, i1, sum_HA, cout_HA);
fullAdder fullAdder_0(i0, i1, cin, sum_FA, cout_FA);
initial begin#6 for(k=0;k<`TESTVECS;k=k+1)
begin #10 {i0, i1, cin}=test_vecs[k]; end
#100 $finish;
end
endmodule
```

OUTPUT




Activities
GTKWave
Apr 28 17:22
GTKWave - hafa.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 186 ns
Marker: 86 ns | Cursor: 85800 ps
SST
tb
Type Signals
reg cin
reg clk
wire cout_FA
wire cout_HA
reg i0
reg i1
integer k
reg reset
wire sum_FA
wire sum_HA
Filter:
Append Insert Replace
Signals
Time
i0 =1
i1 =1
cin =1
sum_FA =1
cout_FA =1
Waves
100 ns

Experiment 3:- Half Subtractor and Full Subtractor

Iverilog Code:-

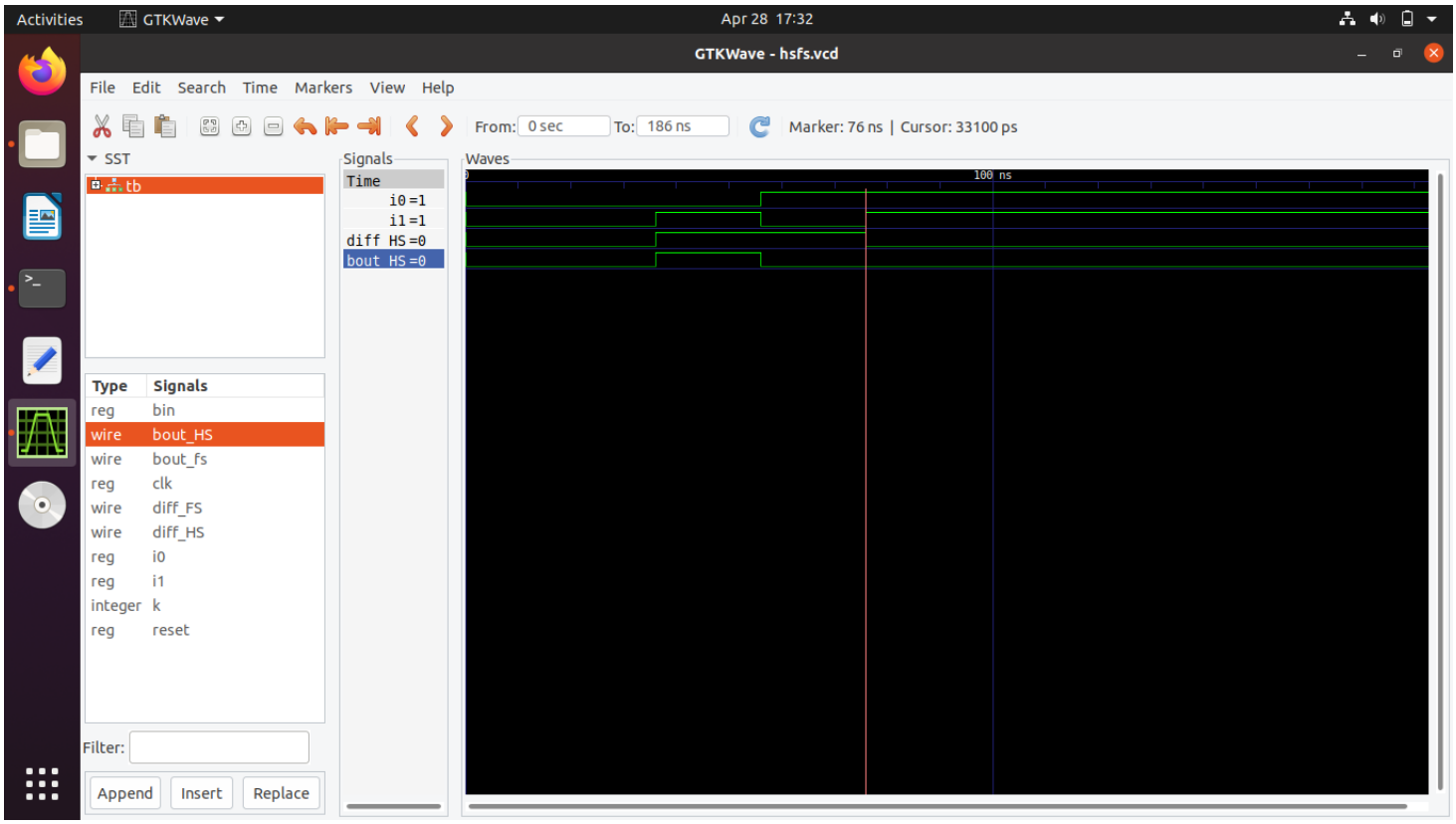
```
module halfSubtractor(input wire a, b, output wire diff, bo);
wire a_bar;
invert invert_0(a, a_bar);
xor2 xor2_0(a, b, diff);
and2 and2_0(a_bar, b, bo);
endmodule

module fullSubtractor(input wire a, b, bin, output wire diff, bo);
wire t1, t2, t3, t4, t5, a_bar;
xor2 xor2_0(a, b, t1);
xor2 xor2_1(t1, bin, diff);
invert invert_0(a, a_bar);
and2 and2_0(a_bar, bin, t2);
and2 and2_1(a_bar, b, t3);
and2 and2_2(b, bin, t4);
or2 or2_0(t2, t3, t5);
or2 or2_1(t5, t4, bo);
endmodule
```

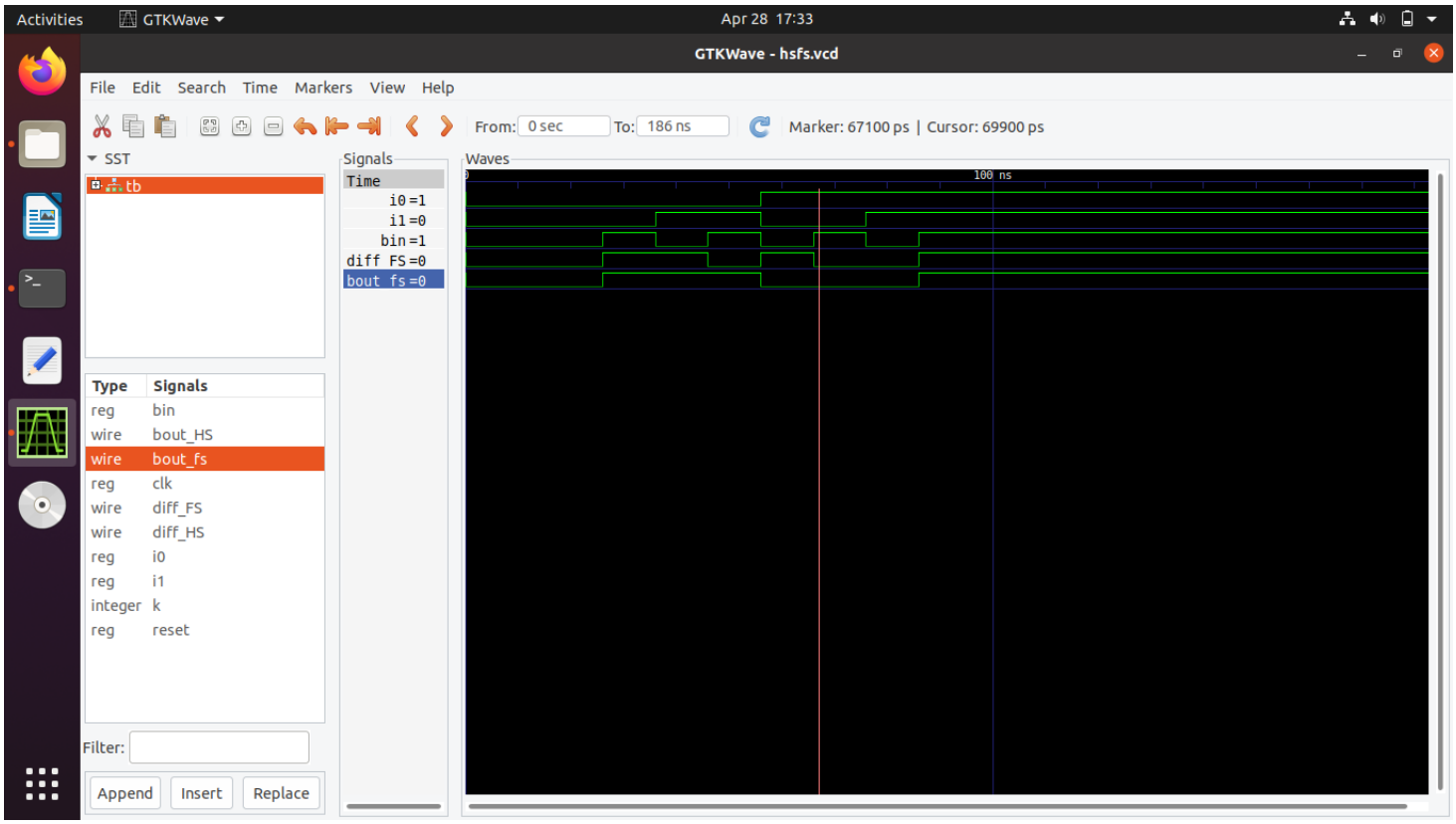
Test Bench:-

```
`timescale 1 ns / 100 ps
`define TESTVECS 8
module tb;
reg clk, reset;
reg i0, i1, bin;
wire diff_HS, bout_HS,diff_FS, bout_fs;
reg [2:0] test_vecs [0:(`TESTVECS-1)];
integer k;
initial begin $dumpfile("hsfs.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] = 3'b000;
test_vecs[1] = 3'b001;
test_vecs[2] = 3'b010;
test_vecs[3] = 3'b011;
test_vecs[4] = 3'b100;
test_vecs[5] = 3'b101;
test_vecs[6] = 3'b110;
test_vecs[7] = 3'b111;
end
initial {i0, i1, bin} = 0;
halfSubtractor halfSubtractor_0(i0, i1, diff_HS, bout_HS);
fullSubtractor fullSubtractor_0(i0, i1, bin, diff_FS, bout_fs);
initial begin
#6 for(k=0;k<`TESTVECS;k=k+1)
begin #10 {i0, i1, bin}=test_vecs[k]; end
#100 $finish;
end
endmodule
```

OUTPUT


Activities
GTKWave
Apr 28 17:32
GTKWave - hsfs.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 186 ns
Marker: 76 ns | Cursor: 33100 ps
SST
tb
Signals
Time
i0 =1
i1 =1
diff HS =0
bout HS =0
Waves
100 ns
Type Signals
reg bin
wire bout_HS
wire bout_fs
reg clk
wire diff_FS
wire diff_HS
reg i0
reg i1
integer k
reg reset
Filter:
Append
Insert
Replace


Activities
GTKWave
Apr 28 17:33
GTKWave - hsfs.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 186 ns
Marker: 67100 ps | Cursor: 69900 ps
SST
tb
Signals
Time
i0 =1
i1 =0
bin =1
diff FS =0
bout fs =0
Waves
100 ns
Type Signals
reg bin
wire bout_HS
wire bout_fs
reg clk
wire diff_FS
wire diff_HS
reg i0
reg i1
integer k
reg reset
Filter:
Append
Insert
Replace

Experiment 4: - 2:1 MUX and 4:1 MUX using 2:1 MUX

Iverilog Code:-

```
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 mux1(i[0],i[1],j0,t0);
mux2 mux2(i[2],i[3],j0,t1);
mux2 mux3(t0,t1,j1,o);
endmodule
```

TESTBENCH[2 to 1 MUX]

```
module TB;
reg A,B,S;
wire X;
initial
begin
$dumpfile("dump.vcd");
$dumpvars(0,TB);
end
mux2 newMUX(.i0(A), .i1(B), .j(S), .o(X));
initial
begin
S = 1'b0;
A = 1'b0;
B = 1'b0;
#5
A = 1'b0;
B = 1'b1;
#5
A = 1'b1;
```

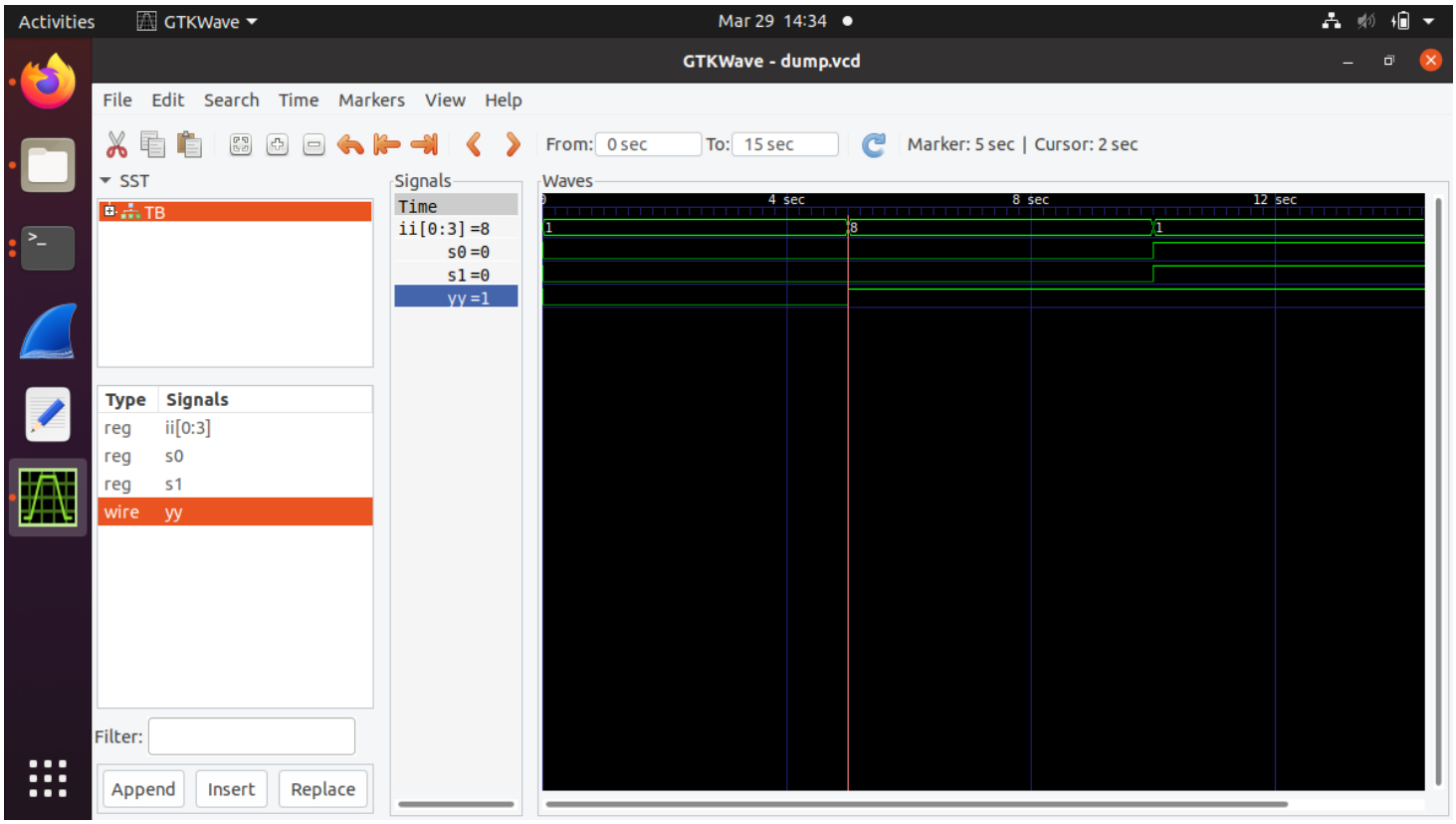
```
B = 1'b0;
#5
A = 1'b1;
B = 1'b1;
#5
S = 1'b0;
A = 1'b0;
B = 1'b0;
#5
A = 1'b0;
B = 1'b1;
#5
A = 1'b1;
B = 1'b0;
#5
A = 1'b1;
B = 1'b1;
end
endmodule
```

TESTBENCH[4 to 1 MUX USING 2 to 1 MUX]

```
module TB;
reg [0:3]ii;
reg s0;
reg s1;
wire yy;
initial
begin
$dumpfile("dump.vcd");
$dumpvars(0, TB);
end
mux4 newMUX(.i(ii), .j0(s0),.j1(s1),.o(yy));
initial
begin
ii = 4'b0001;
```

```
s0=1'b0;
s1=1'b0;
#5
ii = 4'b1000;
#5
ii = 4'b0001;
s0=1'b1;
s1=1'b1;
#5
ii = 4'b0000;
s0=1'b1;
s1=1'b0;
end
endmodule
```

Output:



Experiment 5: - 4 bit Ripple Carry Adder

Verilog Code

//Module full adder.

```
module fulladd(input wire a,b,c, output wire s,cout);
        wire t0,t1,t2,t3,t4;
        xor2 x0(a,b,t0);
        xor2 x1(t0,c,s);
        and2 a0(a,b,t1);
        and2 a1(a,c,t2);
        and2 a2(b,c,t3);
        or2 o1(t1,t2,t4);
        or2 o2(t3,t4,cout);
endmodule
```

// Module 4-bit ripple carry adder.

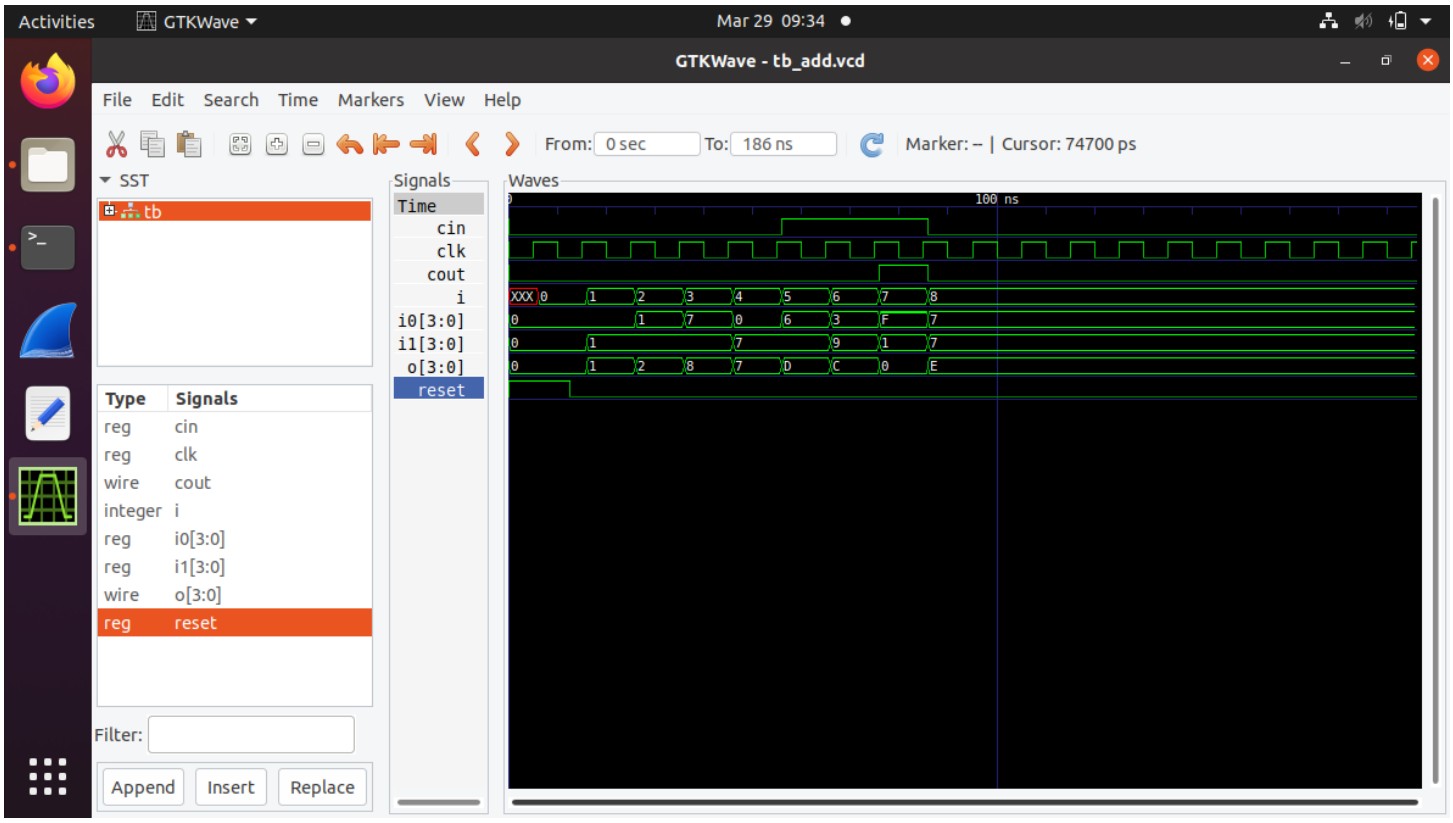
```
module fa4 (input wire [3:0] a, b, input wire cin, output wire [3:0] sum, output wire
cout);
        wire[2:0] t;
        fulladd fa0(a[0],b[0],1'b0,sum[0],t[0]);
        fulladd fa1(a[1],b[1],t[0],sum[1],t[1]);
        fulladd fa2(a[2],b[2],t[1],sum[2],t[2]);
        fulladd fa3(a[3],b[3],t[2],sum[3],cout);
endmodule
```

Test Bench:-

```
`timescale 1 ns / 100 ps
`define TESTVECS 8
module tb;
  reg clk, reset;
  reg [3:0] i0, i1; reg cin;
  wire [3:0] o; wire cout;
  reg [8:0] test_vecs [0:(`TESTVECS-1)];
  integer i;
  initial begin $dumpfile("tb_add.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] = 9'b000000010;
    test_vecs[1] = 9'b000100010;
    test_vecs[2] = 9'b011100010;
    test_vecs[3] = 9'b000001110;
    test_vecs[4] = 9'b011001111;
    test_vecs[5] = 9'b001110011;
    test_vecs[6] = 9'b111100011;
    test_vecs[7] = 9'b011101110;
  end
  initial {i0, i1, cin} = 0;
  fa4 u0 (i0, i1, cin, o, cout);
  initial begin
    #6 for(i=0;i<`TESTVECS;i=i+1)
      begin #10 {i0, i1, cin}=test_vecs[i]; end
    #100 $finish;
  end
endmodule
```

Output:-


Activities
GTKWave
Mar 29 09:34
GTKWave - tb_add.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 186 ns
Marker: -- | Cursor: 74700 ps
SST
tb
Type Signals
reg cin
reg clk
wire cout
integer i
reg i0[3:0]
reg i1[3:0]
wire o[3:0]
reg reset
Filter:
Append Insert Replace
Signals
Time
cin
clk
cout
i
i0[3:0]
i1[3:0]
o[3:0]
reset
Waves
100 ns

Experiment 6: - 16 bit ALU

Verilog Code

```
// Write code for modules you need here

module alu (input wire [1:0] op, input wire [15:0] i0, i1,
   output wire [15:0] o, output wire cout);
        wire c[14:0];
        alu_slice a0(op,i0[0],i1[0],op[0],o[0],c[0]);
        alu_slice a1(op,i0[1],i1[1],c[0],o[1],c[1]);
        alu_slice a2(op,i0[2],i1[2],c[1],o[2],c[2]);
        alu_slice a3(op,i0[3],i1[3],c[2],o[3],c[3]);
        alu_slice a4(op,i0[4],i1[4],c[3],o[4],c[4]);
        alu_slice a5(op,i0[5],i1[5],c[4],o[5],c[5]);
        alu_slice a6(op,i0[6],i1[6],c[5],o[6],c[6]);
        alu_slice a7(op,i0[7],i1[7],c[6],o[7],c[7]);
        alu_slice a8(op,i0[8],i1[8],c[7],o[8],c[8]);
        alu_slice a9(op,i0[9],i1[9],c[8],o[9],c[9]);
        alu_slice a10(op,i0[10],i1[10],c[9],o[10],c[10]);
        alu_slice a11(op,i0[11],i1[11],c[10],o[11],c[11]);
        alu_slice a12(op,i0[12],i1[12],c[11],o[12],c[12]);
        alu_slice a13(op,i0[13],i1[13],c[12],o[13],c[13]);
        alu_slice a14(op,i0[14],i1[14],c[13],o[14],c[14]);
        alu_slice a15(op,i0[15],i1[15],c[14],o[15],cout);
endmodule

// Declare wires here
// Instantiate modules here

module alu_slice(input wire [1:0] op,input wire i0,i1,cin,output wire o,carry);
        wire f1;
        fa16 b0(i0,i1,cin,f1,carry);
        wire t1,t2,t3;
        and2 b1(i0,i1,t1);
        or2 b2(i0,i1,t2);
```

```
        mux2 b3(t1,t2,op[0],t3);
        mux2 b4(t3,f1,op[1],o);
endmodule

module fa16(input wire i0,i1,cin,output wire o,carry);
        wire t4;
        xor2 c1(i1,cin,t4);
        fa c2(i0,t4,cin,o,carry);
endmodule

module fa(input wire a,b,c,output wire s,output wire coutput);
        wire t1;
        xor2 x1(a,b,t1);
        xor2 x2(t1,c,s);
        wire s1;
        and2 y1(a,b,s1);
        wire s2;
        and2 y2(b,c,s2);
        wire s3;
        and2 y3(a,c,s3);
        wire z1;
        or2 x3(s1,s2,z1);
        or2 x4(z1,s3,coutput);
endmodule
```

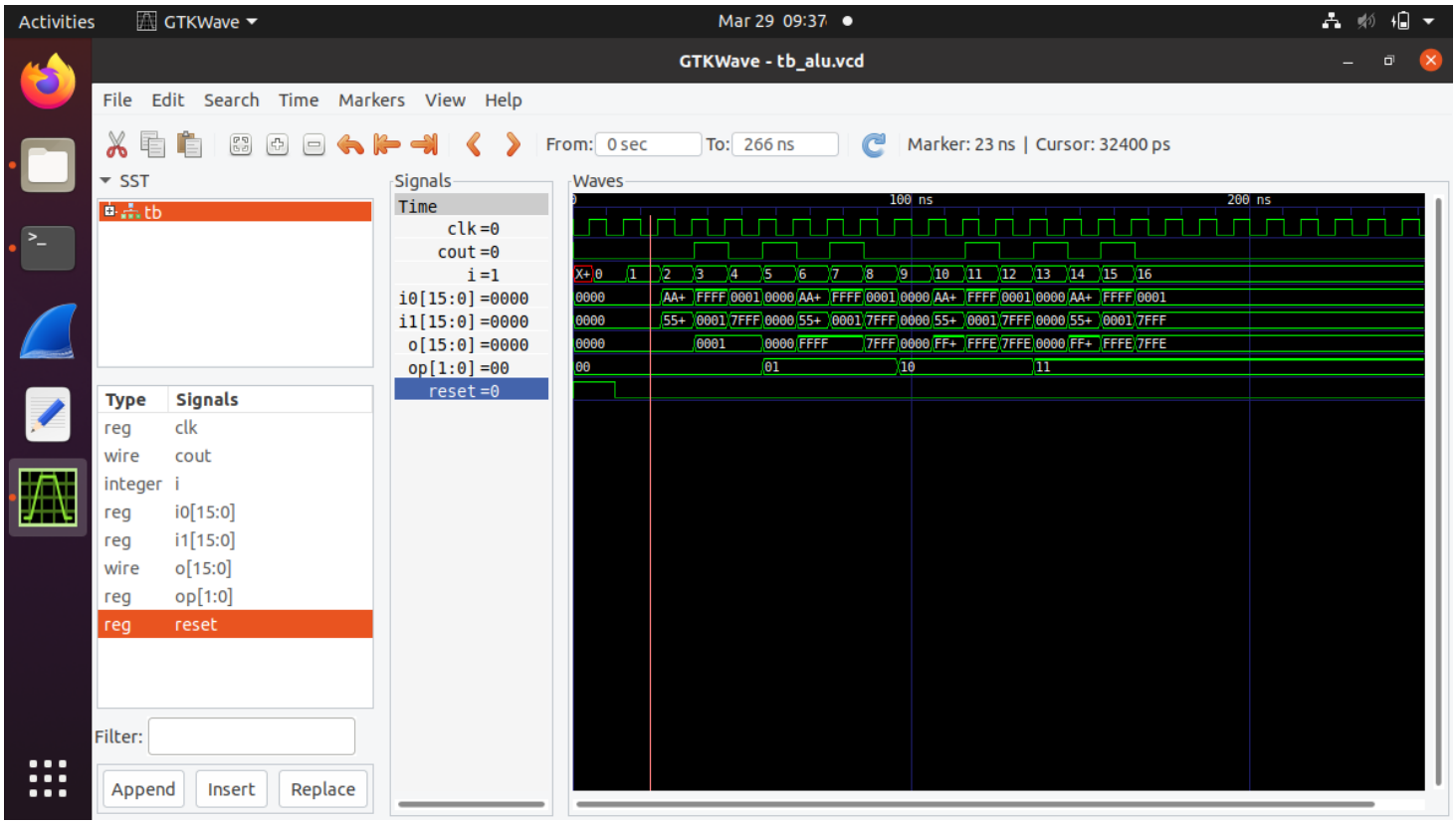
Test Bench:-

```
`timescale 1 ns / 100 ps
`define TESTVECS 16
module tb;
  reg clk, reset;
  reg [1:0] op; reg [15:0] i0, i1;
  wire [15:0] o; wire cout;
  reg [33:0] test_vecs [0:(`TESTVECS-1)];
  integer i;
  initial begin $dumpfile("tb_alu.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][33:32] = 2'b00; test_vecs[0][31:16] = 16'h0000;test_vecs[0][15:0] =
16'h0000;
   test_vecs[1][33:32] = 2'b00; test_vecs[1][31:16] = 16'haa55;test_vecs[1][15:0] =
16'h55aa;
   test_vecs[2][33:32] = 2'b00; test_vecs[2][31:16] = 16'hffff;test_vecs[2][15:0] =
16'h0001;
   test_vecs[3][33:32] = 2'b00; test_vecs[3][31:16] = 16'h0001;test_vecs[3][15:0] =
16'h7fff;
   test_vecs[4][33:32] = 2'b01; test_vecs[4][31:16] = 16'h0000;test_vecs[4][15:0] =
16'h0000;
   test_vecs[5][33:32] = 2'b01; test_vecs[5][31:16] = 16'haa55;test_vecs[5][15:0] =
16'h55aa;
   test_vecs[6][33:32] = 2'b01; test_vecs[6][31:16] = 16'hffff;test_vecs[6][15:0] =
16'h0001;
   test_vecs[7][33:32] = 2'b01; test_vecs[7][31:16] = 16'h0001;test_vecs[7][15:0] =
16'h7fff;
   test_vecs[8][33:32] = 2'b10; test_vecs[8][31:16] = 16'h0000;test_vecs[8][15:0] =
16'h0000;
   test_vecs[9][33:32] = 2'b10; test_vecs[9][31:16] = 16'haa55;test_vecs[9][15:0] =
16'h55aa;
   test_vecs[10][33:32] = 2'b10; test_vecs[10][31:16] = 16'hffff;test_vecs[10][15:0] =
16'h0001;
   test_vecs[11][33:32] = 2'b10; test_vecs[11][31:16] =
16'h0001;test_vecs[11][15:0] = 16'h7fff;
   test_vecs[12][33:32] = 2'b11; test_vecs[12][31:16] =
16'h0000;test_vecs[12][15:0] = 16'h0000;
   test_vecs[13][33:32] = 2'b11; test_vecs[13][31:16] =
16'haa55;test_vecs[13][15:0] = 16'h55aa;
   test_vecs[14][33:32] = 2'b11; test_vecs[14][31:16] = 16'hffff;test_vecs[14][15:0] =
16'h0001;
   test_vecs[15][33:32] = 2'b11; test_vecs[15][31:16] =
16'h0001;test_vecs[15][15:0] = 16'h7fff;
 end
 initial {op, i0, i1} = 0;
 alu alu_0 (op, i0, i1, o, cout);
 initial begin
```

```
   #6 for(i=0;i<`TESTVECS;i=i+1)
     begin #10 {op, i0, i1}=test_vecs[i]; end
   #100 $finish;
  end
endmodule
```

Output:-



Experiment 7: - Register File (of eight 16 bit register)

Verilog Code

```
module mux_4 (input wire i0,i1,i2,i3, input wire j1, j0, output wire o);
  wire  t0, t1;
  mux2 mux2_0 (i0, i1, j1, t0);
  mux2 mux2_1 (i2, i3,j1, t1);
  mux2 mux2_2 (t0, t1, j0, o);
endmodule

module fa(input wire a,b,c,output wire sum,carry);
        wire t1,t2,t3,t4;
        xor2 u0(a,b,t0);
        xor2 u1(c,t0,sum);
        and2 u2(a,b,t1);
        and2 u3(b,c,t2);
        and2 u4(c,a,t3);
        or2 u5(t1,t2,t4);
        or2 u6(t3,t4,carry);
endmodule

module addsub(input wire adsub,i0,i1,cin,output wire sumdiff,cout);
        wire t;
        xor2 a1(i1,adsub,t);
        fa a2(i0,t,cin,sumdiff,cout);
endmodule

module alu_slice(input wire [1:0] op,input wire i0,i1,cin,output wire o,cout);
        wire [3:0]t;
        addsub a1(op[0],i0,i1,cin,t[0],cout);
        and2 a2(i0,i1,t[1]);
        or2 a3(i0,i1,t[2]);
        //mux2 a4(t[1],t[2],op[0],t[3]);
        //mux2 a5(t[0],t[3],op[1],o);
        mux_4 a4(t[0],t[0],t[1],t[2],op[0],op[1],o);
endmodule
```

```
module alu(input wire [1:0] op,input wire [15:0] i0,i1,output wire [15:0] o,output
wire cout);
        wire [14:0]t;
        alu_slice a0(op[1:0],i0[0],i1[0],op[0],o[0],t[0]);
        alu_slice a1(op[1:0],i0[1],i1[1],t[0],o[1],t[1]);
        alu_slice a2(op[1:0],i0[2],i1[2],t[1],o[2],t[2]);
        alu_slice a3(op[1:0],i0[3],i1[3],t[2],o[3],t[3]);
        alu_slice a4(op[1:0],i0[4],i1[4],t[3],o[4],t[4]);
        alu_slice a5(op[1:0],i0[5],i1[5],t[4],o[5],t[5]);
        alu_slice a6(op[1:0],i0[6],i1[6],t[5],o[6],t[6]);
        alu_slice a7(op[1:0],i0[7],i1[7],t[6],o[7],t[7]);
        alu_slice a8(op[1:0],i0[8],i1[8],t[7],o[8],t[8]);
        alu_slice a9(op[1:0],i0[9],i1[9],t[8],o[9],t[9]);
        alu_slice a10(op[1:0],i0[10],i1[10],t[9],o[10],t[10]);
        alu_slice a11(op[1:0],i0[11],i1[11],t[10],o[11],t[11]);
        alu_slice a12(op[1:0],i0[12],i1[12],t[11],o[12],t[12]);
        alu_slice a13(op[1:0],i0[13],i1[13],t[12],o[13],t[13]);
        alu_slice a14(op[1:0],i0[14],i1[14],t[13],o[14],t[14]);
        alu_slice a15(op[1:0],i0[15],i1[15],t[14],o[15],cout);
endmodule
```

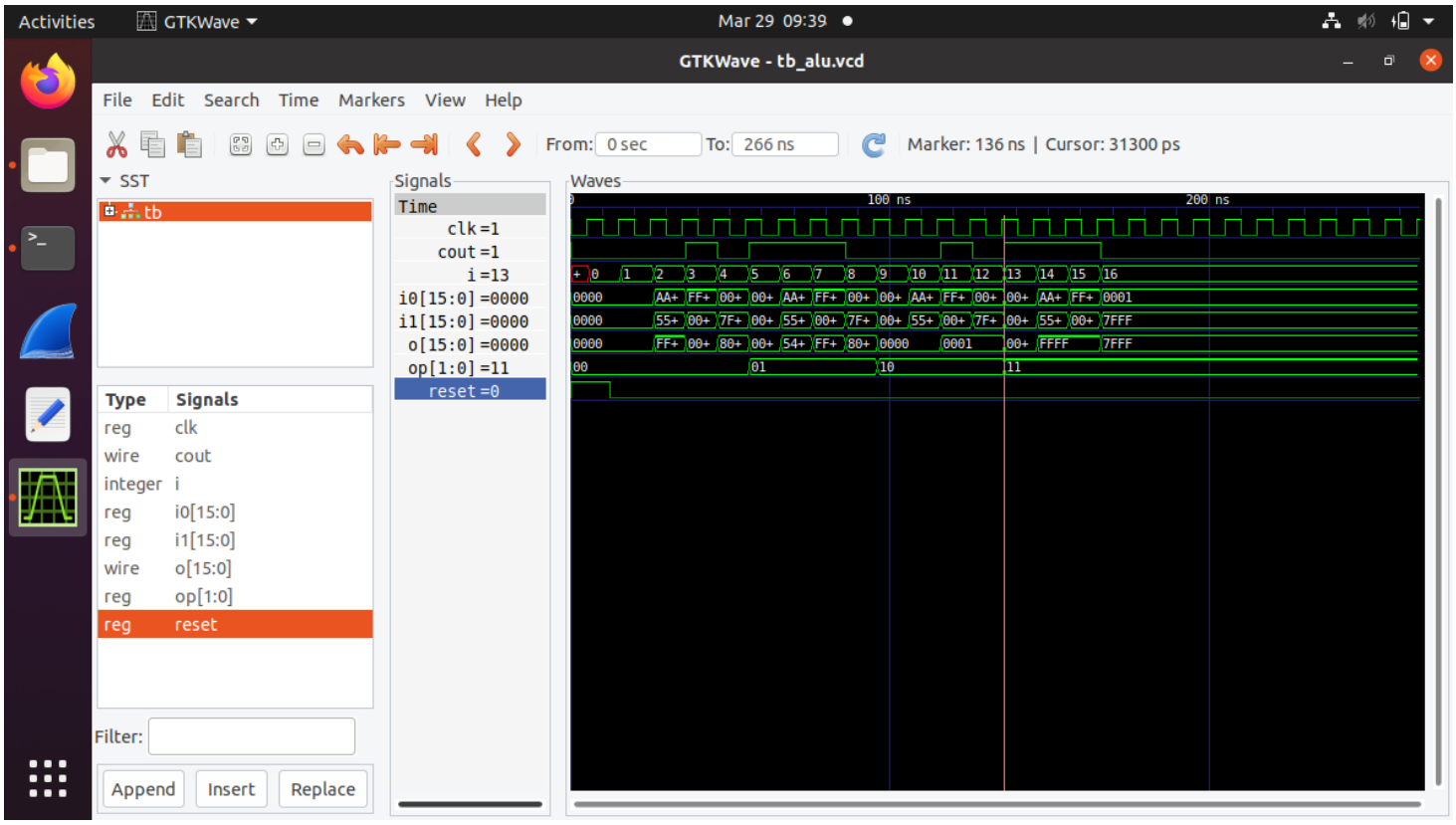
Test Bench:-

```
`timescale 1 ns / 100 ps
`define TESTVECS 16

module tb;
  reg clk, reset;
  reg [1:0] op; reg [15:0] i0, i1;
  wire [15:0] o; wire cout;
  reg [33:0] test_vecs [0:(`TESTVECS-1)];
  integer i;
  initial begin $dumpfile("tb_alu.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][33:32] = 2'b00; test_vecs[0][31:16] = 16'h0000;test_vecs[0][15:0] =
16'h0000;
```

```
   test_vecs[1][33:32] = 2'b00; test_vecs[1][31:16] = 16'haa55;test_vecs[1][15:0] =
16'h55aa;
   test_vecs[2][33:32] = 2'b00; test_vecs[2][31:16] = 16'hffff;test_vecs[2][15:0] =
16'h0001;
   test_vecs[3][33:32] = 2'b00; test_vecs[3][31:16] = 16'h0001;test_vecs[3][15:0] =
16'h7fff;
   test_vecs[4][33:32] = 2'b01; test_vecs[4][31:16] = 16'h0000;test_vecs[4][15:0] =
16'h0000;
   test_vecs[5][33:32] = 2'b01; test_vecs[5][31:16] = 16'haa55;test_vecs[5][15:0] =
16'h55aa;
   test_vecs[6][33:32] = 2'b01; test_vecs[6][31:16] = 16'hffff;test_vecs[6][15:0] =
16'h0001;
   test_vecs[7][33:32] = 2'b01; test_vecs[7][31:16] = 16'h0001;test_vecs[7][15:0] =
16'h7fff;
   test_vecs[8][33:32] = 2'b10; test_vecs[8][31:16] = 16'h0000;test_vecs[8][15:0] =
16'h0000;
   test_vecs[9][33:32] = 2'b10; test_vecs[9][31:16] = 16'haa55;test_vecs[9][15:0] =
16'h55aa;
   test_vecs[10][33:32] = 2'b10; test_vecs[10][31:16] = 16'hffff;test_vecs[10][15:0] =
16'h0001;
   test_vecs[11][33:32] = 2'b10; test_vecs[11][31:16] =
16'h0001;test_vecs[11][15:0] = 16'h7fff;
   test_vecs[12][33:32] = 2'b11; test_vecs[12][31:16] =
16'h0000;test_vecs[12][15:0] = 16'h0000;
   test_vecs[13][33:32] = 2'b11; test_vecs[13][31:16] =
16'haa55;test_vecs[13][15:0] = 16'h55aa;
   test_vecs[14][33:32] = 2'b11; test_vecs[14][31:16] = 16'hffff;test_vecs[14][15:0] =
16'h0001;
   test_vecs[15][33:32] = 2'b11; test_vecs[15][31:16] =
16'h0001;test_vecs[15][15:0] = 16'h7fff;
 end
 initial {op, i0, i1} = 0;
 alu alu_0 (op, i0, i1, o, cout);
 initial begin
   #6 for(i=0;i<`TESTVECS;i=i+1)
     begin #10 {op, i0, i1}=test_vecs[i]; end
   #100 $finish;
```

```
  end
endmodule
```

Output:-



Experiment 8:- Register Integrate with ALU

Iverilog Code:-

```
module dfrl_16 (input wire clk,reset,load, input wire [15:0] din,output wire [15:0]
out);
	dfrl d0(clk,reset,load,din[0],out[0]);
	dfrl d1(clk,reset,load,din[1],out[1]);
	dfrl d2(clk,reset,load,din[2],out[2]);
	dfrl d3(clk,reset,load,din[3],out[3]);
	dfrl d4(clk,reset,load,din[4],out[4]);
	dfrl d5(clk,reset,load,din[5],out[5]);
	dfrl d6(clk,reset,load,din[6],out[6]);
	dfrl d7(clk,reset,load,din[7],out[7]);
	dfrl d8(clk,reset,load,din[8],out[8]);
	dfrl d9(clk,reset,load,din[9],out[9]);
	dfrl d10(clk,reset,load,din[10],out[10]);
	dfrl d11(clk,reset,load,din[11],out[11]);
	dfrl d12(clk,reset,load,din[12],out[12]);
	dfrl d13(clk,reset,load,din[13],out[13]);
	dfrl d14(clk,reset,load,din[14],out[14]);
	dfrl d15(clk,reset,load,din[15],out[15]);
endmodule

module mux128_16(input wire[15:0] i0,i1,i2,i3,i4,i5,i6,i7, input wire[2:0] j, output
wire[15:0] o);
	mux8 m0({i0[0],i1[0],i2[0],i3[0],i4[0],i5[0],i6[0],i7[0]},j[0],j[1],j[2],o[0]);
	mux8 m1({i0[1],i1[1],i2[1],i3[1],i4[1],i5[1],i6[1],i7[1]},j[0],j[1],j[2],o[1]);
	mux8 m2({i0[2],i1[2],i2[2],i3[2],i4[2],i5[2],i6[2],i7[2]},j[0],j[1],j[2],o[2]);
	mux8 m3({i0[3],i1[3],i2[3],i3[3],i4[3],i5[3],i6[3],i7[3]},j[0],j[1],j[2],o[3]);
	mux8 m4({i0[4],i1[4],i2[4],i3[4],i4[4],i5[4],i6[4],i7[4]},j[0],j[1],j[2],o[4]);
	mux8 m5({i0[5],i1[5],i2[5],i3[5],i4[5],i5[5],i6[5],i7[5]},j[0],j[1],j[2],o[5]);
	mux8 m6({i0[6],i1[6],i2[6],i3[6],i4[6],i5[6],i6[6],i7[6]},j[0],j[1],j[2],o[6]);
	mux8 m7({i0[7],i1[7],i2[7],i3[7],i4[7],i5[7],i6[7],i7[7]},j[0],j[1],j[2],o[7]);
	mux8 m8({i0[8],i1[8],i2[8],i3[8],i4[8],i5[8],i6[8],i7[8]},j[0],j[1],j[2],o[8]);
	mux8 m9({i0[9],i1[9],i2[9],i3[9],i4[9],i5[9],i6[9],i7[9]},j[0],j[1],j[2],o[9]);
```

```
        mux8
m10({i0[10],i1[10],i2[10],i3[10],i4[10],i5[10],i6[10],i7[10]},j[0],j[1],j[2],o[10]);
        mux8
m11({i0[11],i1[11],i2[11],i3[11],i4[11],i5[11],i6[11],i7[11]},j[0],j[1],j[2],o[11]);
        mux8
m12({i0[12],i1[12],i2[12],i3[12],i4[12],i5[12],i6[12],i7[12]},j[0],j[1],j[2],o[12]);
        mux8
m13({i0[13],i1[13],i2[13],i3[13],i4[13],i5[13],i6[13],i7[13]},j[0],j[1],j[2],o[13]);
        mux8
m14({i0[14],i1[14],i2[14],i3[14],i4[14],i5[14],i6[14],i7[14]},j[0],j[1],j[2],o[14]);
        mux8
m15({i0[15],i1[15],i2[15],i3[15],i4[15],i5[15],i6[15],i7[15]},j[0],j[1],j[2],o[15]);
endmodule


module reg_file (input wire clk, reset, wr, input wire [2:0] rd_addr_a, rd_addr_b,
wr_addr, input wire [15:0] d_in, output wire [15:0] d_out_a, d_out_b);
        wire [0:7] load;
        wire [0:15] dout0, dout1, dout2, dout3, dout4, dout5, dout6, dout7;
        dfrl_16 df0(clk, reset, load[0], d_in, dout0);
        dfrl_16 df1(clk, reset, load[1], d_in, dout1);
        dfrl_16 df2(clk, reset, load[2], d_in, dout2);
        dfrl_16 df3(clk, reset, load[3], d_in, dout3);
        dfrl_16 df4(clk, reset, load[4], d_in, dout4);
        dfrl_16 df5(clk, reset, load[5], d_in, dout5);
        dfrl_16 dfr6(clk, reset, load[6], d_in, dout6);
        dfrl_16 df7(clk, reset, load[7], d_in, dout7);
        demux8 demux0(wr, wr_addr[2], wr_addr[1], wr_addr[0],load);
        mux128_16 mux9(dout0, dout1, dout2, dout3, dout4, dout5, dout6, dout7,
        rd_addr_a, d_out_a);
        mux128_16 mux10(dout0, dout1, dout2, dout3, dout4, dout5, dout6, dout7,
rd_addr_b, d_out_b);
          endmodule

module mux2_16(input wire [15:0] din_regular, alu_out, input wire j, output wire
[15:0]din_final);
        mux2 m0(din_regular[0], alu_out[0],j, din_final[0]);
```

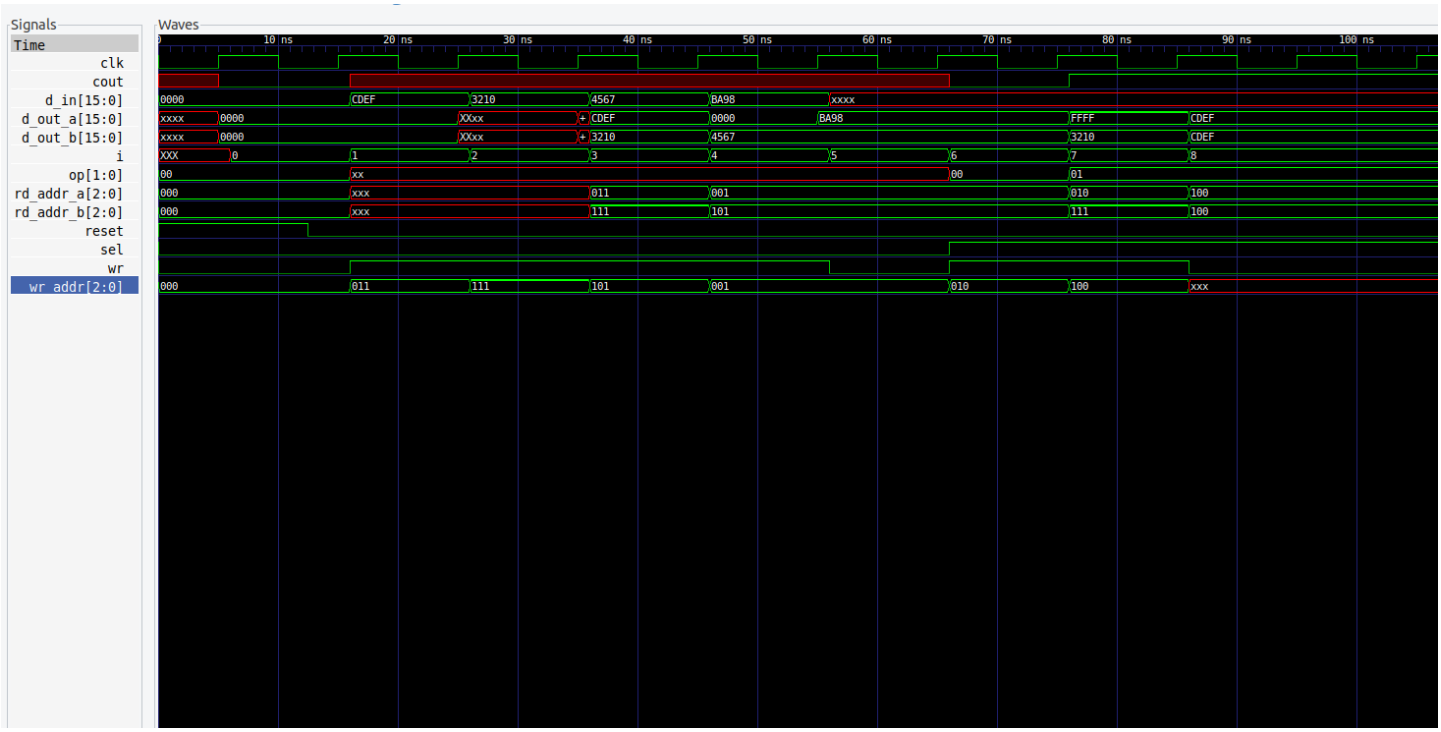
```
        mux2 m1(din_regular[1], alu_out[1], j, din_final[1]);
        mux2 m2(din_regular[2], alu_out[2],j, din_final[2]);
        mux2 m3(din_regular[3], alu_out[3], j, din_final[3]);
        mux2 m4(din_regular[4], alu_out[4], j, din_final[4]);
        mux2 m5(din_regular[5], alu_out[5], j, din_final[5]);
        mux2 m6(din_regular[6], alu_out[6], j, din_final[6]);
        mux2 m7(din_regular[7], alu_out[7], j, din_final[7]);
        mux2 m8(din_regular[8], alu_out[8], j, din_final[8]);
        mux2 m9(din_regular[9], alu_out[9], j, din_final[9]);
        mux2 m10(din_regular[10], alu_out[10], j, din_final[10]);
        mux2 m11(din_regular[11], alu_out[11], j, din_final[11]);
        mux2 m12(din_regular[12], alu_out[12], j, din_final[12]);
        mux2 m13(din_regular[13], alu_out[13], j, din_final[13]);
        mux2 m14(din_regular[14], alu_out[14], j, din_final[14]);
        mux2 m15(din_regular[15], alu_out[15], j, din_final[15]);
endmodule

module reg_alu (input wire clk, reset, sel, wr, input wire [1:0] op, input wire [2:0]
rd_addr_a,
          rd_addr_b, wr_addr, input wire [15:0] d_in, output wire [15:0] d_out_a,
d_out_b, output wire )
        wire [15:0] din_alu;
        wire [15:0] din_reg;
        mux2_16 m1(d_in, din_alu, sel, din_reg);
        reg_file reg1(clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, din_reg, d_out_a,
d_out_b);
        alu a1(op, d_out_a, d_out_b, din_alu, cout);

endmodule
`timescale 1 ns / 100 ps
```

Test Bench Code:-

```
`define TESTVECS 8

module tb;
 reg clk, reset, wr, sel;
 reg [1:0] op;
 reg [2:0] rd_addr_a, rd_addr_b, wr_addr; reg [15:0] d_in;
 wire [15:0] d_out_a, d_out_b;
 reg [28:0] test_vecs [0:(`TESTVECS-1)];
 integer i;
 initial begin $dumpfile("tb_reg_alu.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][28] = 1'b0; test_vecs[0][27] = 1'b1; test_vecs[0][26:25] = 2'bxx;
  test_vecs[0][24:22] = 3'ox; test_vecs[0][21:19] = 3'ox;
  test_vecs[0][18:16] = 3'h3; test_vecs[0][15:0] = 16'hcdef;

  test_vecs[1][28] = 1'b0; test_vecs[1][27] = 1'b1; test_vecs[1][26:25] = 2'bxx;
  test_vecs[1][24:22] = 3'ox; test_vecs[1][21:19] = 3'ox;
  test_vecs[1][18:16] = 3'o7; test_vecs[1][15:0] = 16'h3210;

  test_vecs[2][28] = 1'b0; test_vecs[2][27] = 1'b1; test_vecs[2][26:25] = 2'bxx;
  test_vecs[2][24:22] = 3'o3; test_vecs[2][21:19] = 3'o7;
  test_vecs[2][18:16] = 3'o5; test_vecs[2][15:0] = 16'h4567;

  test_vecs[3][28] = 1'b0; test_vecs[3][27] = 1'b1; test_vecs[3][26:25] = 2'bxx;
  test_vecs[3][24:22] = 3'o1; test_vecs[3][21:19] = 3'o5;
  test_vecs[3][18:16] = 3'o1; test_vecs[3][15:0] = 16'hba98;

  test_vecs[4][28] = 1'b0; test_vecs[4][27] = 1'b0; test_vecs[4][26:25] = 2'bxx;
  test_vecs[4][24:22] = 3'o1; test_vecs[4][21:19] = 3'o5;
  test_vecs[4][18:16] = 3'o1; test_vecs[4][15:0] = 16'hxxxx;

  test_vecs[5][28] = 1'b1; test_vecs[5][27] = 1'b1; test_vecs[5][26:25] = 2'b00;
  test_vecs[5][24:22] = 3'o1; test_vecs[5][21:19] = 3'o5;
```

```
    test_vecs[5][18:16] = 3'o2; test_vecs[5][15:0] = 16'hxxxx;

    test_vecs[6][28] = 1'b1; test_vecs[6][27] = 1'b1; test_vecs[6][26:25] = 2'b01;
    test_vecs[6][24:22] = 3'o2; test_vecs[6][21:19] = 3'o7;
    test_vecs[6][18:16] = 3'o4; test_vecs[6][15:0] = 16'hxxxx;

    test_vecs[7][28] = 1'b1; test_vecs[7][27] = 1'b0; test_vecs[7][26:25] = 2'b01;
    test_vecs[7][24:22] = 3'o4; test_vecs[7][21:19] = 3'o4;
    test_vecs[7][18:16] = 3'ox; test_vecs[7][15:0] = 16'hxxxx;
  end
  initial {sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in} = 0;
  reg_alu reg_alu_0 (clk, reset, sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in,
  d_out_a, d_out_b, cout);
  initial begin
    #6 for(i=0;i<`TESTVECS;i=i+1)
      begin #10 {sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in}=test_vecs[i]; end
    #100 $finish;
  end
endmodule
```

Output:-



Experiment 9:- Program Counter

Iverilog Code:-

```
// Write code for modules you need here
module fadd(input wire a, b, cin, output wire sum, cout);
wire t0,t1,t2,t3;
xor2 x0(a,b,t0);
xor2 x1(t0,cin,sum);
and2 x2(a,b,t1);
and2 x3(b,cin,t2);
and2 x4(cin,a,t3);
or3 x5(t1,t2,t3,cout);
endmodule

module addsub(input wire mode,i0,i1,cin,output wire sumdiff,cout);
wire t;
xor2 x0(i1,mode,t);
fadd f(i0,t,cin,sumdiff,cout);
endmodule

module pc_slice0 (input wire clk,reset,offset,inc,sub,load, output wire pc,cout);
wire t1,t2;
or2 o0(offset,inc,t1);
addsub o2(sub,pc,t1,sub,t2,cout);
dfrl o3(clk,reset,load,t2,pc);
endmodule

module pc_slice1 (input wire clk,reset,offset,inc,sub,load,cin, output wire pc,cout);
wire t1,t2,t3;
invert i(inc,t1);
and2 o0(offset,t1,t2);
addsub o2(sub,pc,t2,cin,t3,cout);
dfrl o3(clk,reset,load,t3,pc);
endmodule
```

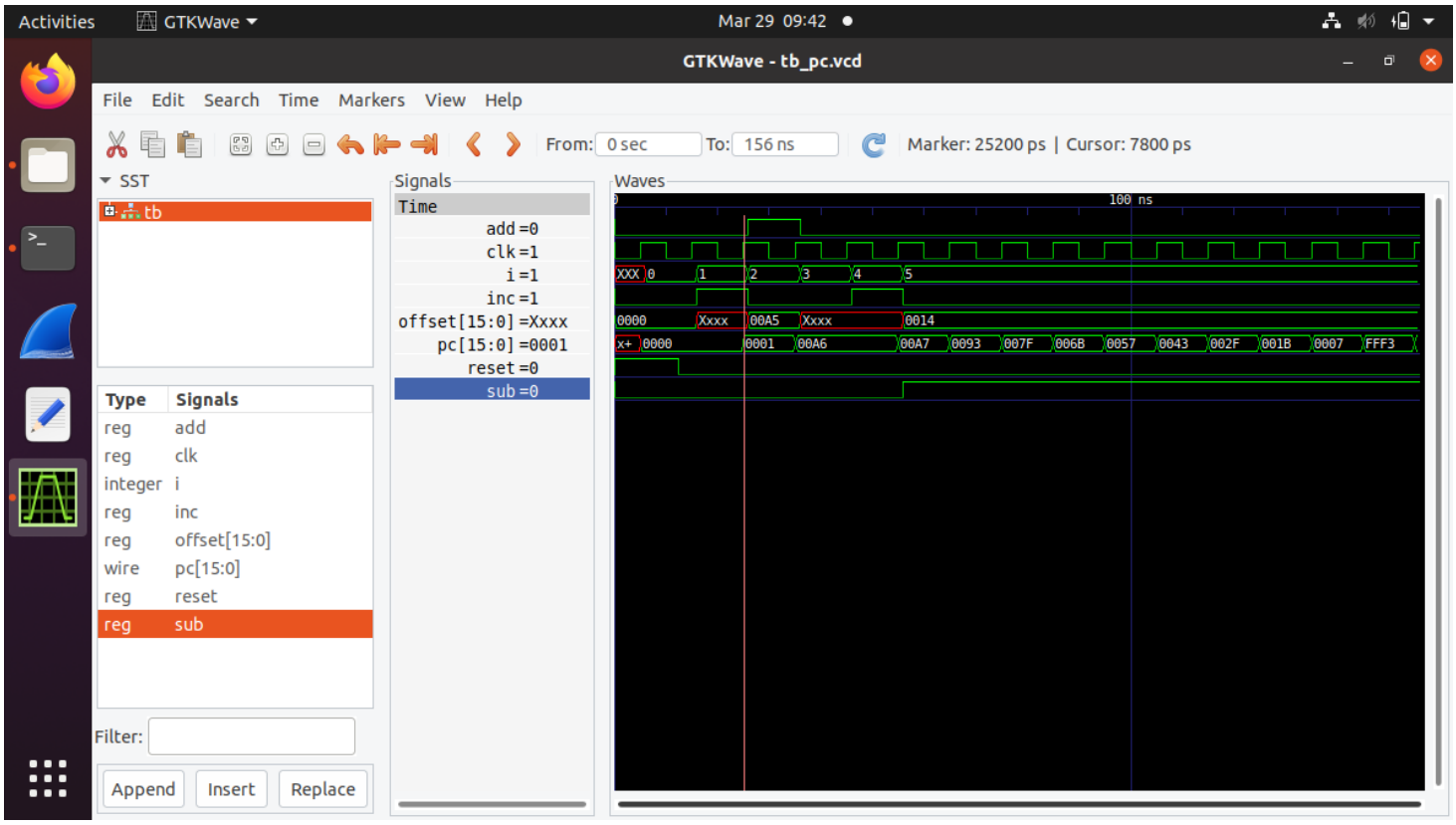
```
module pc (input wire clk, reset, inc, add, sub, input wire [15:0] offset, output wire
[15:0] pc);
wire[15:0]cout;
wire load;
or3 o(inc,add,sub,load);
pc_slice0 s0(clk,reset,offset[0],inc,sub,load,pc[0],cout[0]);
pc_slice1 s1(clk,reset,offset[1],inc,sub,load,cout[0],pc[1],cout[1]);
pc_slice1 s2(clk,reset,offset[2],inc,sub,load,cout[1],pc[2],cout[2]);
pc_slice1 s3(clk,reset,offset[3],inc,sub,load,cout[2],pc[3],cout[3]);
pc_slice1 s4(clk,reset,offset[4],inc,sub,load,cout[3],pc[4],cout[4]);
pc_slice1 s5(clk,reset,offset[5],inc,sub,load,cout[4],pc[5],cout[5]);
pc_slice1 s6(clk,reset,offset[6],inc,sub,load,cout[5],pc[6],cout[6]);
pc_slice1 s7(clk,reset,offset[7],inc,sub,load,cout[6],pc[7],cout[7]);
pc_slice1 s8(clk,reset,offset[8],inc,sub,load,cout[7],pc[8],cout[8]);
pc_slice1 s9(clk,reset,offset[9],inc,sub,load,cout[8],pc[9],cout[9]);
pc_slice1 s10(clk,reset,offset[10],inc,sub,load,cout[9],pc[10],cout[10]);
pc_slice1 s11(clk,reset,offset[11],inc,sub,load,cout[10],pc[11],cout[11]);
pc_slice1 s12(clk,reset,offset[12],inc,sub,load,cout[11],pc[12],cout[12]);
pc_slice1 s13(clk,reset,offset[13],inc,sub,load,cout[12],pc[13],cout[13]);
pc_slice1 s14(clk,reset,offset[14],inc,sub,load,cout[13],pc[14],cout[14]);
pc_slice1 s15(clk,reset,offset[15],inc,sub,load,cout[14],pc[15],cout[15]);

endmodule
```

Test Bench:-

```
`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 :-


Activities
GTKWave
Mar 29 09:42
GTKWave - tb_pc.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 156 ns
Marker: 25200 ps | Cursor: 7800 ps
SST
tb
Type Signals
reg add
reg clk
integer i
reg inc
reg offset[15:0]
wire pc[15:0]
reg reset
reg sub
Filter:
Append Insert Replace
Signals
Time
add =0
clk =1
i =1
inc =1
offset[15:0] =Xxxx
pc[15:0] =0001
reset =0
sub =0
Waves
100 ns