

A  
Practical Activity Report  
Submitted for

# **Embedded Systems Design**

## **(Verilog Programming)**



**Submitted to:**  
Vivek Mehta  
Res. Scholar  
CSED, TIET

**Submitted by:**  
COE-10 Group

**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**  
**THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY, PATIALA-147004,**  
**PUNJAB**  
**INDIA**  
***Jan-June 2019***

**Ques 1: WAP to implement all logic gates (AND, OR, NAND, NOR, XOR, NOT) using data flow modelling. Also verify the Truth Table and generate Timewave.**

**Ans:**

```
module logic_gates(a,b,c,d,e,f,g,h);
input a,b;
output c,d,e,f,g,h;
assign c=a&b;
assign d=a|b;
assign e=a~&b;
assign f=a~|b;
assign g=a^b;
assign h=~a;
endmodule

module test_bench;
reg a,b;
wire c,d,e,f,g,h;
logic_gates t(a,b,c,d,e,f,g,h);
initial begin
$dumpfile("h_dm.vcd");
$dumpvars(0,t);
$monitor("a=%b,b=%b,and=%b,or=%b,nand=%b,nor=%b,xor=
%b,not=%b",a,b,c,d,e,f,g,h);
a=0;    b=0;    #10
a=1;    b=0;    #10
a=0;    b=1;    #10
a=1;    b=1;    #10
end
```

```
$finish;  
end  
endmodule
```

OUTPUT:

```
a=0,b=0,and=0,or=0,nand=1,nor=1,xor=0,not=1  
a=1,b=0,and=0,or=1,nand=1,nor=0,xor=1,not=0  
a=0,b=1,and=0,or=1,nand=1,nor=0,xor=1,not=1  
a=1,b=1,and=1,or=1,nand=0,nor=0,xor=0,not=0
```

TRUTH TABLE



TIMEWAVE

**Ques 2 (a): WAP to implement Half adder. Also verify the Truth Table and generate Timewave.**

**Ans:**

```
module halfadder(num1,num2,sum,carry);  
    input num1, num2;  
    output sum, carry;  
    assign sum = num1^num2;  
    assign carry = num1&num2;  
endmodule
```

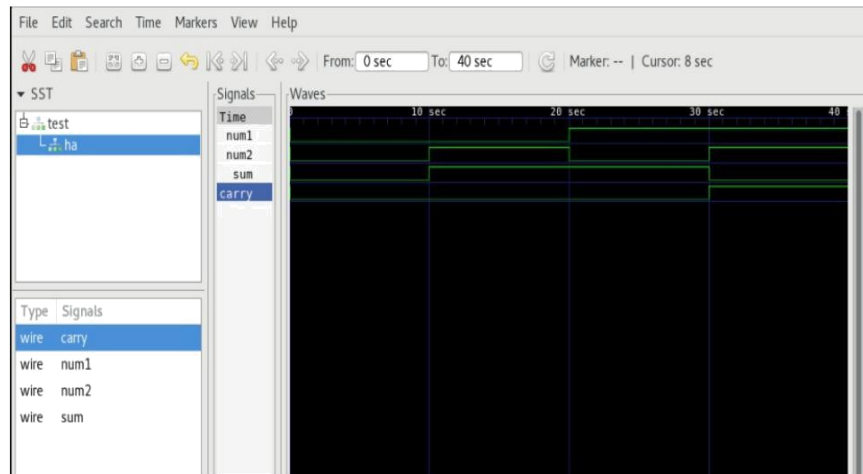
```
module test;  
    reg a,b;  
    wire c,d;  
    halfadder ha(a,b,c,d);  
    initial begin  
        $display("a b c d");  
        $dumpfile("halfadder_dataflow.vcd");  
        $dumpvars(0,test);  
        $monitor("%b %b %b %b", a,b,c,d);  
        a=0;  b=0;  #10  
        a=0;  b=1;  #10  
        a=1;  b=0;  #10  
        a=1;  b=1;  #10  $finish;  
    end
```

endmodule

OUTPUT:

```
File Edit View Search Terminal Help
[CS@localhost CML1_006]$ iverilog halfadder_dataflow1.v
[CS@localhost CML1_006]$ vvp a.out
a b c d
VCD info: dumpfile halfadder_dataflow1.vcd opened for output.
0 0 0 0
0 1 1 0
1 0 1 0
1 1 0 1
[CS@localhost CML1_006]$ gtkwave halfadder_dataflow1.vcd
```

TRUTH TABLE



TIMEWAVE

**Ques2 (b): WAP to implement Full adder. Also verify the Truth Table and generate Timewave.**

**Ans:**

```
module fulladder(num1,num2,num3,sum,carry);  
    input num1, num2, num3;  
    output sum, carry;  
    assign sum = num1^num2^num3;  
    assign carry = (num1&num2)|  
(num2&num3)|(num1&num3);  
endmodule  
  
module test;  
    reg a,b,c;  
    wire d,e;  
    fulladder fa(a,b,c,d,e);  
    initial begin  
        $display("a b c          d e");  
        $dumpfile("fulladder_dataflow.vcd");  
        $dumpvars(0,test);  
        $monitor("%b %b %b          %b %b", a,b,c,d,e);  
        a=0;  b=0;  c=0;  #10
```

```

a=0;  b=0;  c=1;  #10
a=0;  b=1;  c=0;  #10
a=0;  b=1;  c=1;  #10
a=1;  b=0;  c=0;  #10
a=1;  b=0;  c=1;  #10
a=1;  b=1;  c=0;  #10
a=1;  b=1;  c=1;  #10 $finish;
end

```

endmodule

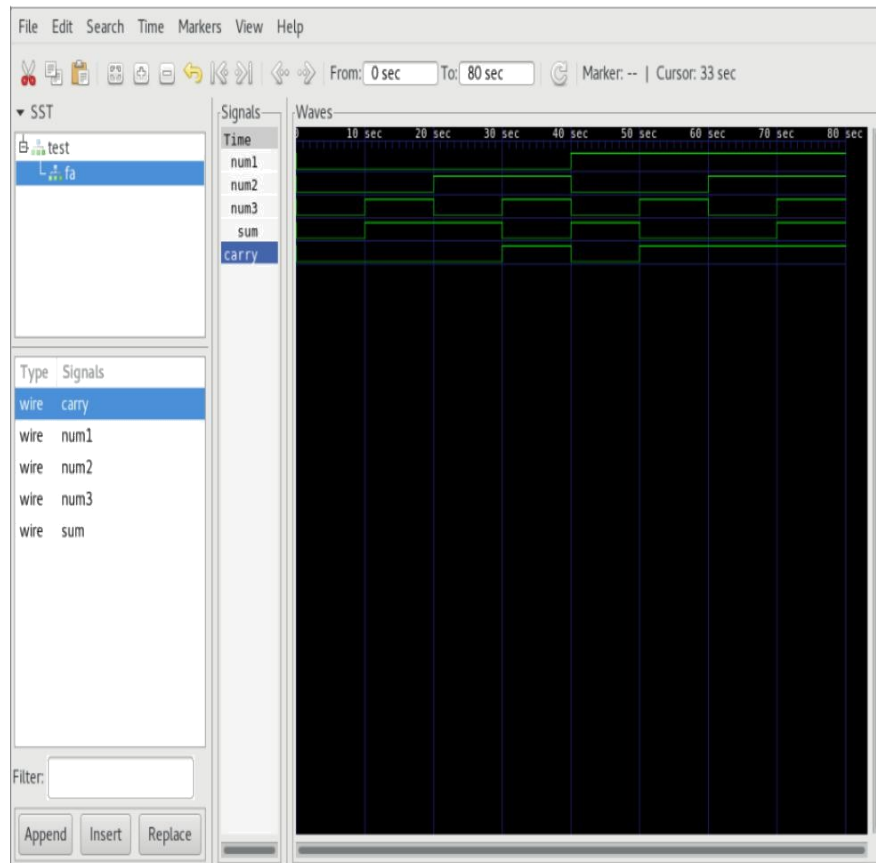
OUTPUT:

```

File Edit View Search Terminal Help
[CSSED@localdomain CML1_006]$ iverilog fulladder_dataflo
w.v
[CSSED@localdomain CML1_006]$ vvp a.out
a b c          d e
VCD info: dumpfile fulladder_dataflow.vcd opened for ou
tput.
0 0 0          0 0
0 0 1          1 0
0 1 0          1 0
0 1 1          0 1
1 0 0          1 0
1 0 1          0 1
1 1 0          0 1
1 1 1          1 1
[CSSED@localdomain CML1_006]$ █

```

TRUTH TABLE



TIMEWAVE



**Ques3 (a): WAP to implement Half Subtractor. Also verify the Truth Table and generate Timewave.**

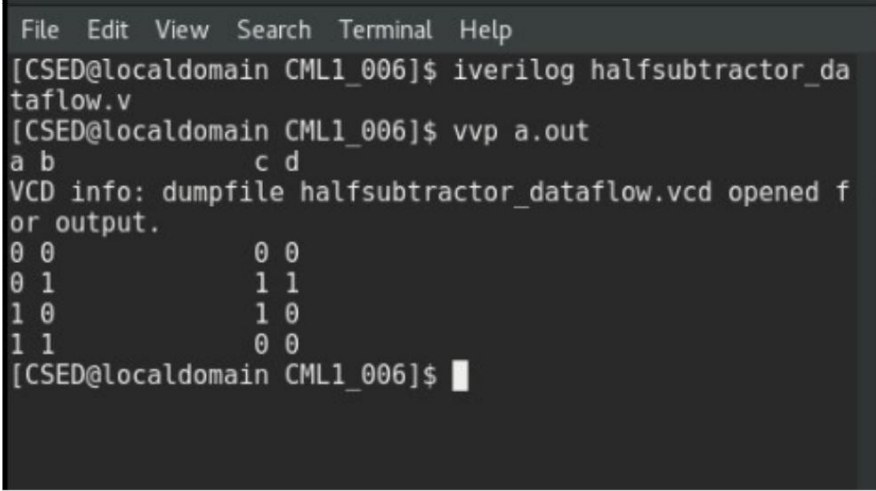
**Ans:**

```
module halfsubtractor(num1,num2,diff,borrow);  
    input num1, num2;  
    output diff, borrow;  
    wire r1;  
    assign diff = num1^num2;  
    assign r1 = ~num1;  
    assign borrow = r1&num2;  
endmodule
```

```
module test;  
    reg a,b;  
    wire c,d;  
    halfsubtractor hs(a,b,c,d);  
    initial begin  
        $display("a b      c d");  
        $dumpfile("halfsubtractor_dataflow.vcd");  
        $dumpvars(0,test);  
        $monitor("%b %b      %b %b", a,b,c,d);  
        a=0;  b=0;  #10
```

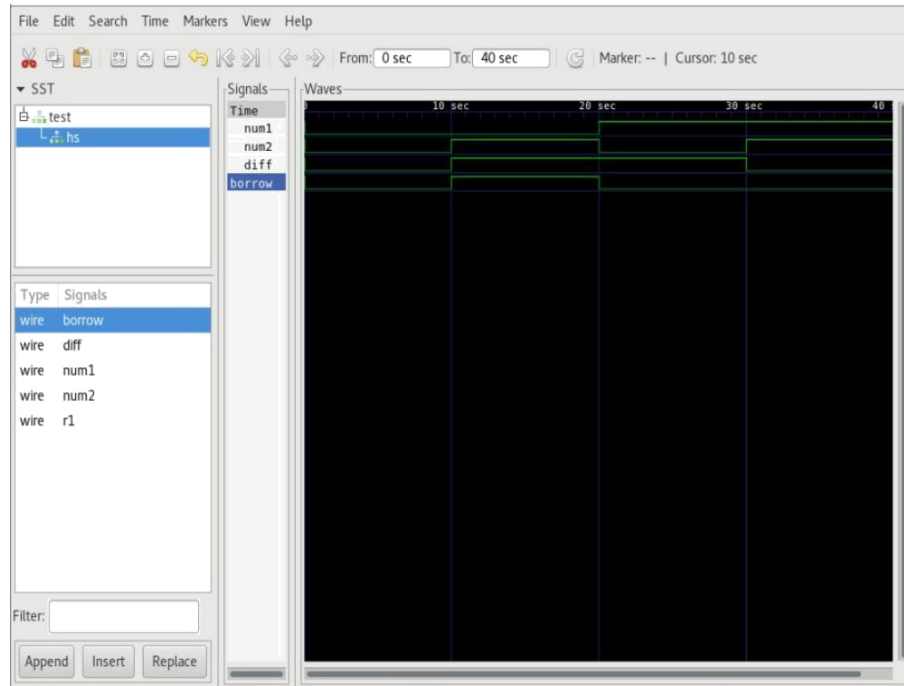
```
a=0; b=1; #10
a=1; b=0; #10
a=1; b=1; #10 $finish;
end
endmodule
```

OUTPUT:



```
File Edit View Search Terminal Help
[CS@localdomain CML1_006]$ iverilog halfsubtractor_dataflow.v
[CS@localdomain CML1_006]$ vvp a.out
a b          c d
VCD info: dumpfile halfsubtractor_dataflow.vcd opened for output.
0 0          0 0
0 1          1 1
1 0          1 0
1 1          0 0
[CS@localdomain CML1_006]$
```

TRUTH TABLE



TIMEWAVE

**Ques 3 (b): Implementation of Full subtractor. Verify logic using truth table and time wave**

**Ans:**

```
module fullsubtractor(num1,num2,num3,diff,borrow);  
    input num1, num2, num3;  
    output diff, borrow;  
    wire r1, r2, r3, r4, r5;  
    assign r1 = ~num1;  
    assign r2 = ~num2;  
    assign r3 = ~num3;  
    assign diff = (r1&r2&num3)|(r1&num2&r3)|  
(num1&r2&r3)|(num1&num2&num3);  
    assign borrow = (r1&num3)|(r1&num2)|(num2&num3);  
endmodule  
  
module test;  
    reg a,b,c;  
    wire d,e;  
    fullsubtractor fs(a,b,c,d,e);  
    initial begin  
        $display("a b c          d e");  
        $dumpfile("fullsubtractor_dataflow.vcd");  
        $dumpvars(0,test);  
    end  
endmodule
```

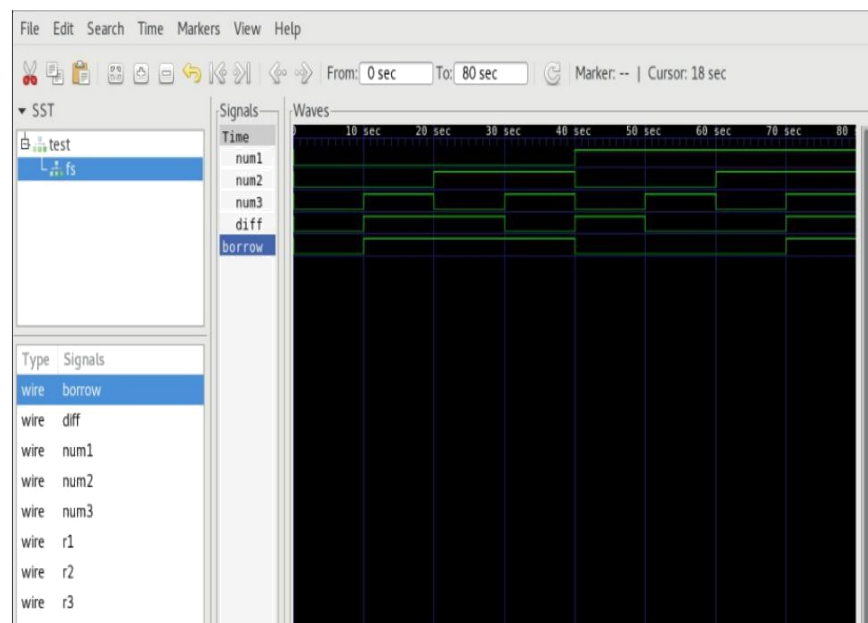
```
$monitor("%b %b %b          %b %b", a,b,c,d,e);
a=0;  b=0;  c=0;   #10
a=0;  b=0;  c=1;   #10
a=0;  b=1;  c=0;   #10
a=0;  b=1;  c=1;   #10
a=1;  b=0;  c=0;   #10
a=1;  b=0;  c=1;   #10
a=1;  b=1;  c=0;   #10
a=1;  b=1;  c=1;   #10 $finish;
end

endmodule
```

OUTPUT:

```
File Edit View Search Terminal Help
[CS@localdomain CML1_006]$ iverilog fullsubtractor.v
[CS@localdomain CML1_006]$ vvp a.out
a b c          d e
VCD info: dumpfile fullsubtractor_dataflow.vcd opened for output.
0 0 0          0 0
0 0 1          1 1
0 1 0          1 1
0 1 1          0 1
1 0 0          1 0
1 0 1          0 0
1 1 0          0 0
1 1 1          1 1
[CS@localdomain CML1_006]$
```

TRUTH TABLE



TIMEWAVE

**Ques 4: WAP to implement 4x2 Encoder using Case statement and take input and output lines as vector . Also verify the Truth Table and generate Timewave.**

**Ans:**

```
module encoder(inp,out);  
input [3:0]inp;  
output [1:0]out;  
reg [1:0]out;  
always@(*)begin  
out=2'b00;  
case(inp)  
4'b0001:out=2'b00;  
4'b0010:out=2'b01;  
4'b0100:out=2'b10;  
4'b1000:out=2'b11;  
endcase  
end  
endmodule
```

```
module test;  
reg [3:0]inp;  
wire [1:0]out;  
encoder t(inp,out);
```

```
initial begin
$dumpfile("h9.vcd");
$dumpvars(0,test);
$display("input\tout");
$monitor("%b\t%b",inp,out);
inp=4'b0001; #10
inp=4'b0010; #10
inp=4'b0100; #10
inp=4'b1000; #10
$finish;
end
endmodule
```

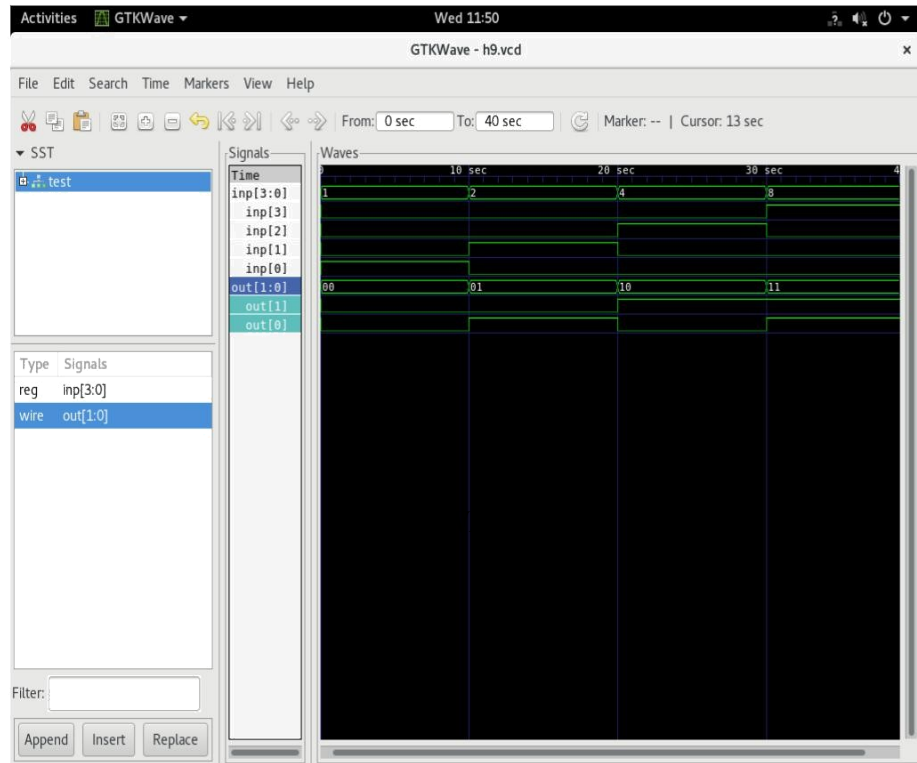
OUTPUT:



```
Activities Terminal Wed 11:38
CSED@localdomain:~
File Edit View Search Terminal Help
[CSED@localdomain ~]$ iverilog pro9.v
[CSED@localdomain ~]$ vvp ./a.out
VCD info: dumpfile h.vcd opened for output.
input  out
0001   00
0010   01
0100   10
1000   11
[CSED@localdomain ~]$
```

TRUTH TABLE





**Ques 5: WAP to implement 3x8 Decoder using Case statement and take input and output lines as vector. Also verify the Truth Table and generate Timewave.**

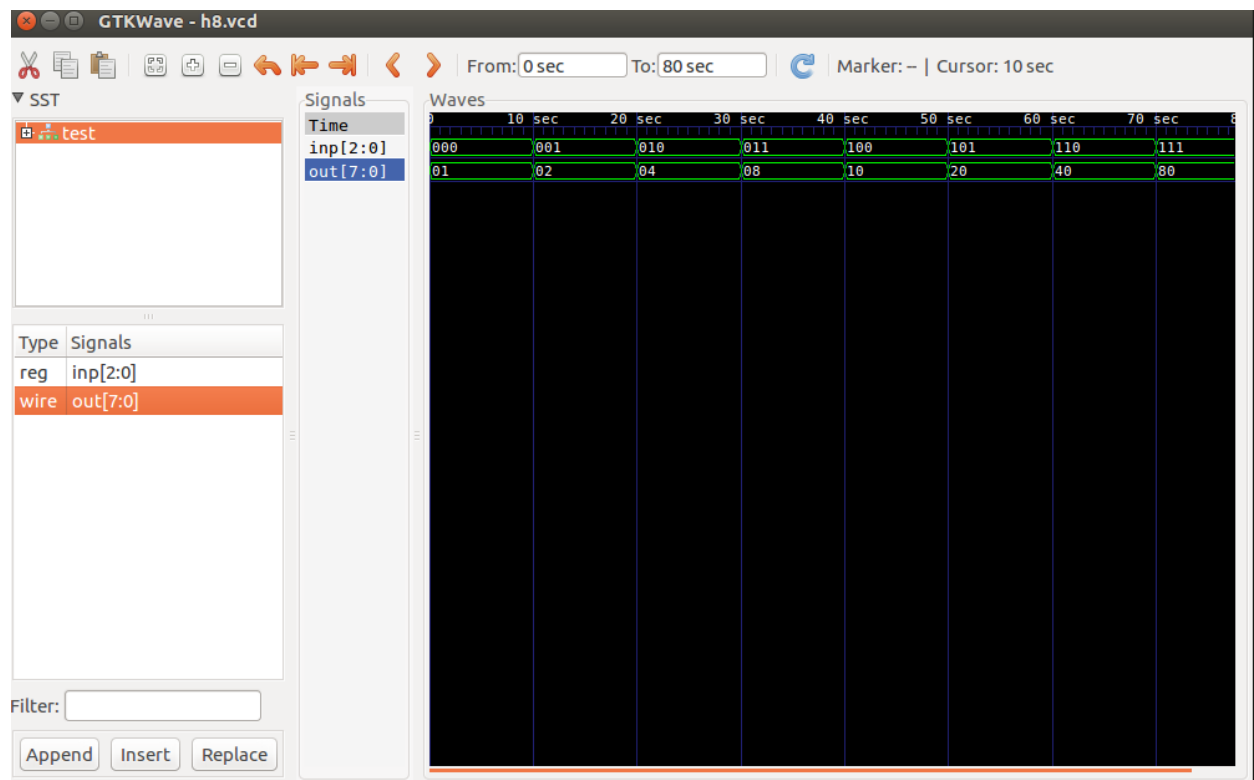
**Ans:**

```
module decoder(inp,out);  
input [2:0]inp;  
output [7:0]out;  
reg [7:0]out;  
always@(*)begin  
case(inp)  
3'b000:out=8'b00000001;  
3'b001:out=8'b00000010;  
3'b010:out=8'b00000100;  
3'b011:out=8'b00001000;  
3'b100:out=8'b00010000;  
3'b101:out=8'b00100000;  
3'b110:out=8'b01000000;  
3'b111:out=8'b10000000;  
endcase  
end  
endmodule  
  
module test;
```

```
reg [2:0]inp;
wire [7:0]out;
decoder t(inp,out);
initial begin
$dumpfile("h8.vcd");
$dumpvars(0,test);
$display("input\tout");
$monitor("%b\t%b",inp,out);
inp=3'b000;  #10
inp=3'b001;  #10
inp=3'b010;  #10
inp=3'b011;  #10
inp=3'b100;  #10
inp=3'b101;  #10
inp=3'b110;  #10
inp=3'b111;  #10
$finish;
end
endmodule
OUTPUT:
```

```
jasvir@jasvir-HP-Notebook: ~/Desktop
jasvir@jasvir-HP-Notebook:~/Desktop$ cd..
cd..: command not found
jasvir@jasvir-HP-Notebook:~/Desktop$ iverilog decoder.v
jasvir@jasvir-HP-Notebook:~/Desktop$ vvp.out
vvp.out: command not found
jasvir@jasvir-HP-Notebook:~/Desktop$ vvp a.out
VCD info: dumpfile h8.vcd opened for output.
input    out
000      00000001
001      00000010
010      00000100
011      00001000
100      00010000
101      00100000
110      01000000
111      10000000
```

TRUTH TABLE



GTK WAVE

**Ques 6 : Write a program to implement 2x1 MUX using conditional operator. Also verify the Truth Table and generate Timewave.**

**Ans:**

```
module mux(inp1,inp2,sel,out);
input inp1,inp2,sel;
output out;
assign out=sel?inp2:inp1;
endmodule

module test;
reg a,b,c;
wire o;
mux t(.sel(a),.inp1(b),.inp2(c),.out(o));
initial begin
$dumpfile("h5.vcd");
$dumpvars(0,test);
$display("select\tinput1\tinput2\toutput");
$monitor("%b\t%b\t%b\t%b",a,b,c,o);
a=0;  b=0;  c=0;  #10
a=0;  b=0;  c=1;  #10
a=0;  b=1;  c=0;  #10
a=0;  b=1;  c=1;  #10
```

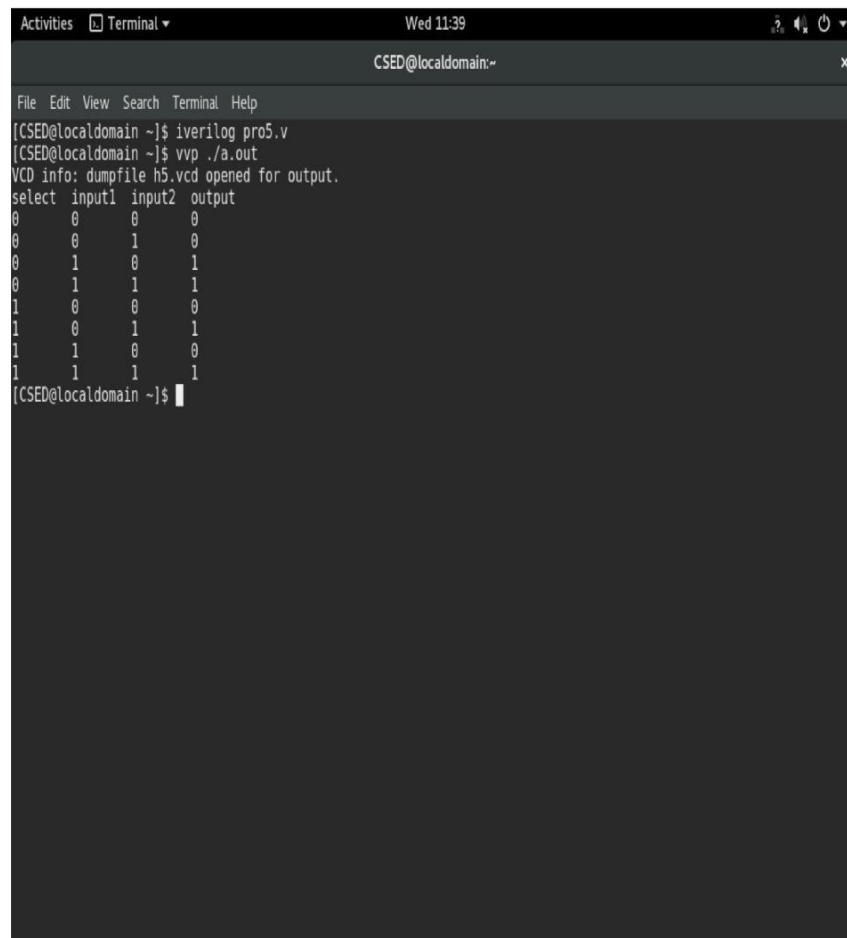
```
a=1;  b=0;  c=0;  #10
a=1;  b=0;  c=1;  #10
a=1;  b=1;  c=0;  #10
a=1;  b=1;  c=1;  #10

$finish;

end

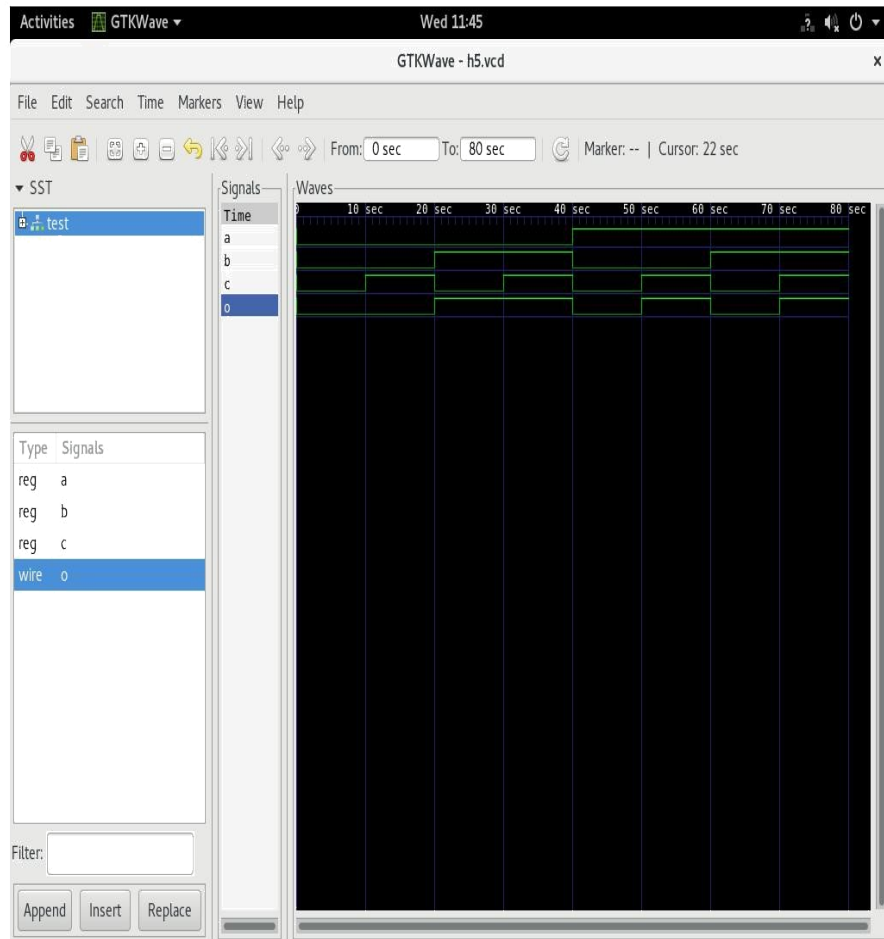
endmodule
```

OUTPUT:



```
Activities Terminal Wed 11:39
CSED@localdomain:~
File Edit View Search Terminal Help
[CSED@localdomain ~]$ iverilog pro5.v
[CSED@localdomain ~]$ vvp ./a.out
VCD info: dumpfile h5.vcd opened for output.
select input1 input2 output
0 0 0 0
0 0 1 0
0 1 0 1
0 1 1 1
1 0 0 0
1 0 1 1
1 1 0 0
1 1 1 1
[CSED@localdomain ~]$
```

TRUTH TABLE



TIMEWAVE



**Ques7: Write a program to implement 1x4 DEMUX using if else-if statement  
Also verify the Truth Table and generate Timewave.**

**Ans:**

```
module demux(inp,sel,out);  
input inp;  
input [1:0]sel;  
output [3:0]out;  
reg [3:0]out;  
always@(*)begin  
out[0]=0;  
out[1]=0;  
out[2]=0;  
out[3]=0;  
if (sel==2'b00)begin  
out[0]=inp;  
end  
else if (sel==2'b01)begin  
out[1]=inp;  
end  
else if (sel==2'b10)begin  
out[2]=inp;  
end  
else begin  
out[3]=inp;  
end  
end
```

```

end

endmodule

module test;

reg inp;

reg [1:0]sel;

wire [3:0]out;

demux t(inp,sel,out);

initial begin

$dumpfile("h.vcd");

$dumpvars(0,test);

$display("input\tsel1\tsel0\to0\to1\to2\to3");

$monitor("%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t",inp,sel[1],sel[0],out[0],out[1],out[2],out[3]);

inp=0;      sel[0]=0;      sel[1]=0;      #10
inp=0;      sel[0]=0;      sel[1]=1;      #10
inp=0;      sel[0]=1;      sel[1]=0;      #10
inp=0;      sel[0]=1;      sel[1]=1;      #10
inp=1;      sel[0]=0;      sel[1]=0;      #10
inp=1;      sel[0]=0;      sel[1]=1;      #10
inp=1;      sel[0]=1;      sel[1]=0;      #10
inp=1;      sel[0]=1;      sel[1]=1;      #10

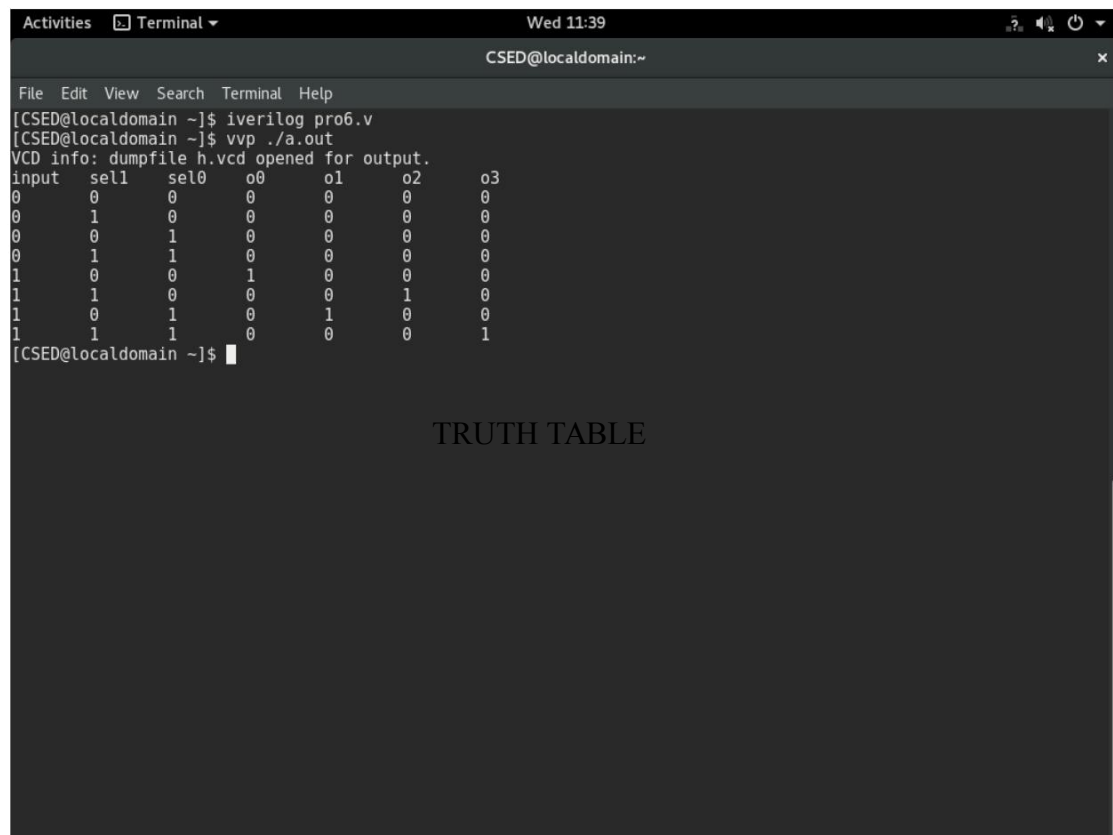
$finish;

end

```

endmodule

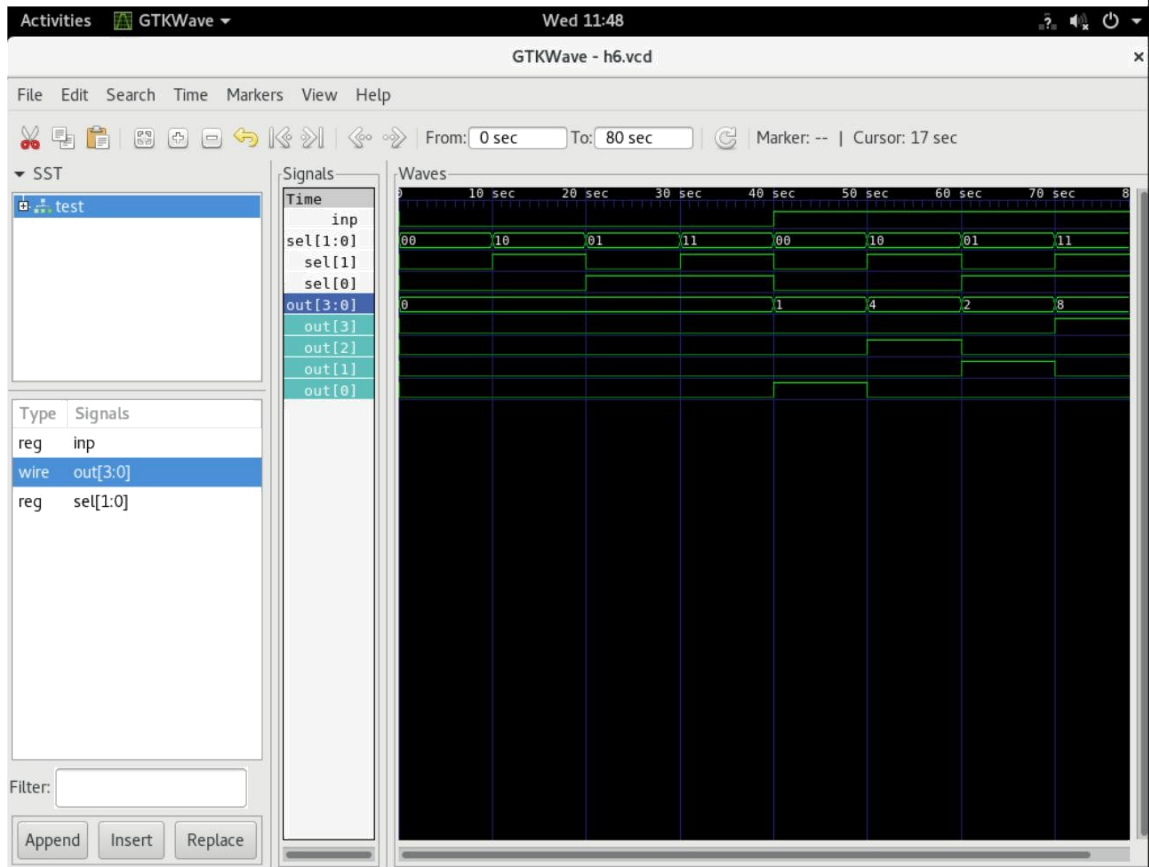
OUTPUT:



The screenshot shows a terminal window titled "CSED@localdomain:~" with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (Wed 11:39). The terminal displays the following text:

```
[CSED@localdomain ~]$ iverilog pro6.v
[CSED@localdomain ~]$ vvp ./a.out
VCD info: dumpfile h.vcd opened for output.
input  sel1  sel0  o0    o1    o2    o3
0      0     0     0     0     0     0
0      1     0     0     0     0     0
0      0     1     0     0     0     0
0      1     1     0     0     0     0
1      0     0     1     0     0     0
1      1     0     0     0     1     0
1      0     1     0     1     0     0
1      1     1     0     0     0     1
[CSED@localdomain ~]$
```

Below the terminal output, the text "TRUTH TABLE" is displayed.



**TIMEWAVE**

**Ques 8:a) WAP to implement SR Flip Flop with an input clock using Case statement and take input and output lines as vector . Also verify the Truth Table and generate Timewave.**

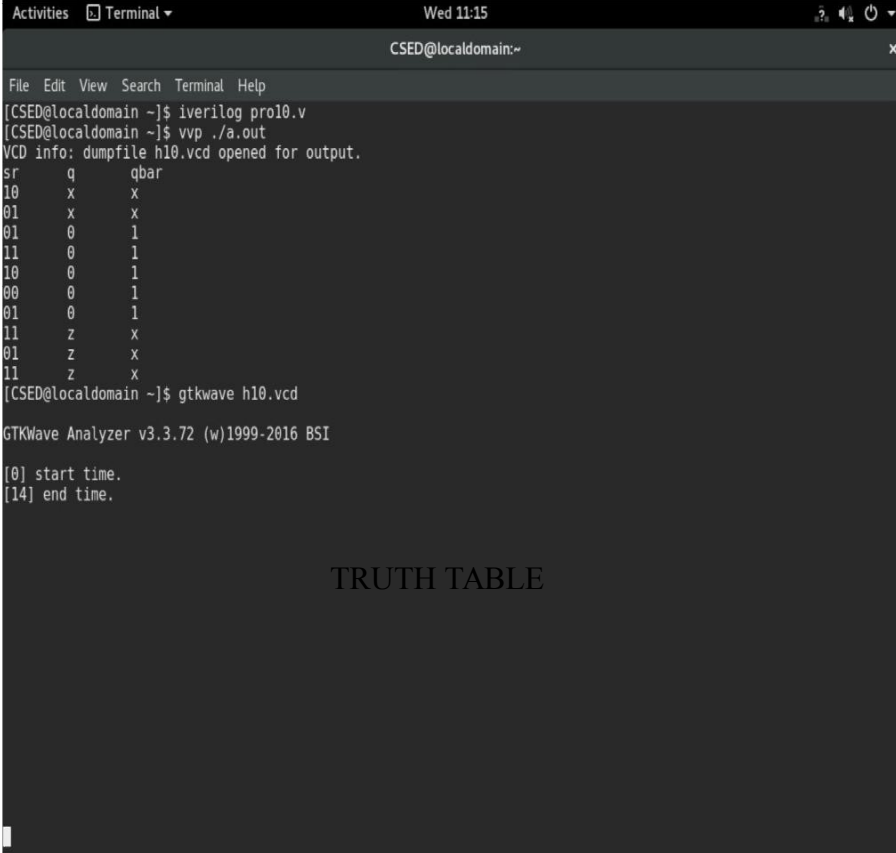
**Ans:**

```
module srff(input [1:0]sr, input clk,output q,output qbar);  
    reg q,qbar;  
    integer i;  
    always @(posedge clk) begin  
        case(sr)  
            2'b00: q=q;  
            2'b01: q=0;  
            2'b10: q=1;  
            2'b11: q=1'bz;  
        endcase  
        qbar=~q;  
    end  
endmodule  
  
module test;  
    reg [1:0]sr;  
    reg clk;  
    wire q, qbar;  
    integer i;
```

```
srff t(sr,clk,q,qbar);  
initial begin  
$dumpfile("h10.vcd");  
$dumpvars(0,test);  
$display("sr\tq\tqbar");  
$monitor("%b\t%b\t%b",sr,q,qbar);  
sr=2'b10; #1  
sr=2'b01; #2  
sr=2'b11; #1  
sr=2'b10; #1  
sr=2'b00; #3  
sr=2'b01; #2  
sr=2'b11; #1  
sr=2'b01; #2  
sr=2'b11; #1  
$finish;  
end  
initial begin  
clk=0;  
for (i=0;i<30;i++)  
#2 clk=~clk;  
end
```

endmodule

OUTPUT:



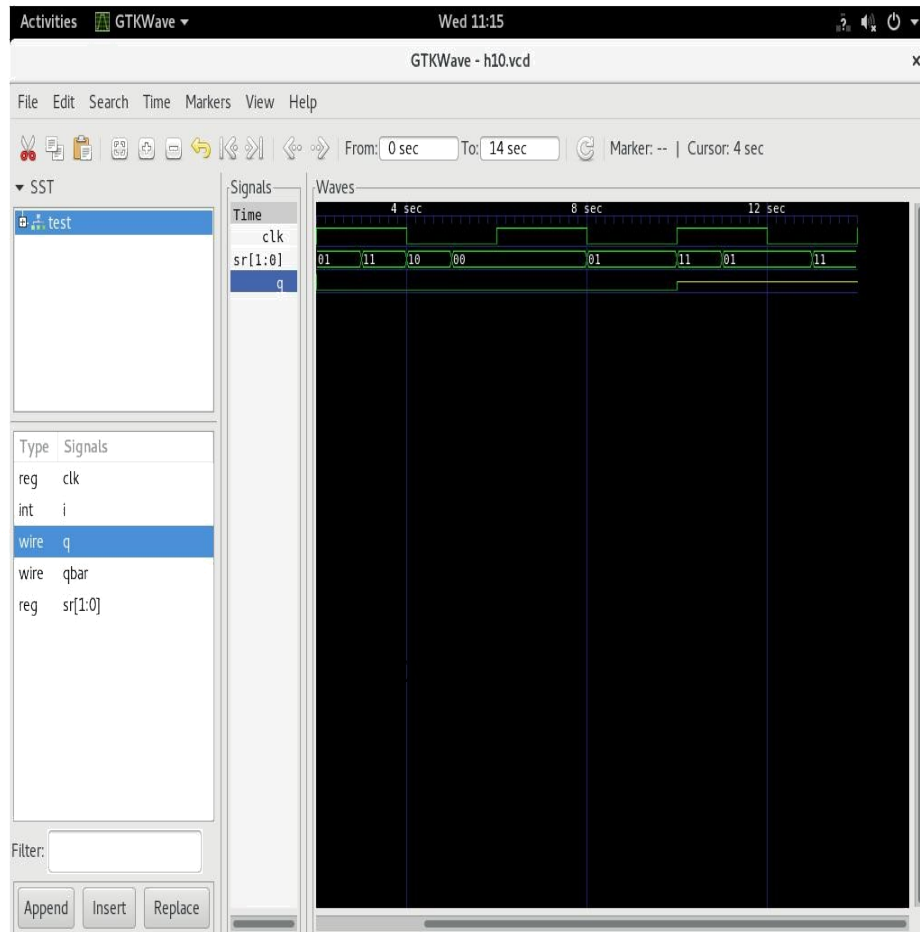
The screenshot shows a terminal window titled 'CSED@localdomain:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output is as follows:

```
[CSED@localdomain ~]$ iverilog pro10.v
[CSED@localdomain ~]$ vvp ./a.out
VCD info: dumpfile h10.vcd opened for output.
sr      q      qbar
10      x      x
01      x      x
01      0      1
11      0      1
10      0      1
00      0      1
01      0      1
11      z      x
01      z      x
11      z      x
[CSED@localdomain ~]$ gtkwave h10.vcd

GTKWave Analyzer v3.3.72 (w)1999-2016 BSI

[0] start time.
[14] end time.
```

TRUTH TABLE



Q 1).Write a program in Verilog to implement D Flip Flop using positive edge triggered clock.Verify logic using truth table and time wave.?



```

module synD(d,q, qbar, clk, reset);
input d, clk, reset;

output q, qbar;

reg q, qbar;

always@(posedge clk) begin

q=d;

if(reset==1) begin

q=0;
end

qbar= ~q;

end

endmodule

```

```

module test;

reg d,clk, reset;

wire q, qbar;

integer i;

synD t(d,q, qbar, clk, reset);

initial begin

$dumpfile("h1.vcd");

$dumpvars(0, test);

$display("D\t Q \t Qbar \tReset");

$monitor("%b\t%b\t%b\t%b", d, q, qbar, reset);

d=0; reset=1; #3

d=1; reset=1; #3

d=0; reset=0; #5

d=0; reset=1; #3

d=1; reset=0; #3

d=0; reset=1; #5

```

```
$finish;  
end  
  
initial begin  
    clk=0; reset=1;  
    for(i=0; i<50; i++) begin  
        #3 clk= ~clk;  
    end  
end  
  
endmodule
```

Command Prompt

```
Microsoft Windows [Version 10.0.17134.407]  
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Users\DELL>cd C:\iverilog\bin
```

```
C:\iverilog\bin>iverilog dflipflop.v
```

```
C:\iverilog\bin>vvp a.out
```

```
VCD info: dumpfile h1.vcd opened for output.
```

D	Q	Qbar	Reset
0	x	x	1
1	0	1	1
0	0	1	0
0	0	1	1
1	0	1	0
1	1	0	0
0	1	0	1
0	0	1	1

```
C:\iverilog\bin>
```

0 (a) Write a program in Verilog to implement T Flip Flop using positive edge triggered clock. Verify logic using truth table and time wave.

```

module synT(t,q, qbar, clk, reset);
input t, clk, reset;
output q, qbar;
reg q, qbar;
always@(posedge clk) begin
    q=~t;
    if(reset==1) begin
        q=0;
    end
    qbar= ~q;
end
endmodule

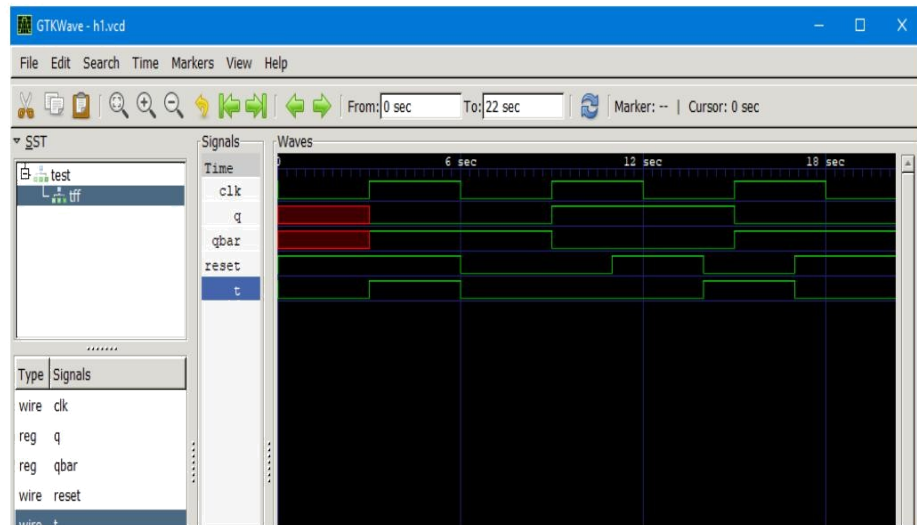
```

```

module test;
reg d,clk, reset;
wire q, qbar;
integer i;
synT(t,q, qbar, clk, reset);
initial begin
    $dumpfile("h1.vcd");
    $dumpvars(0, test);
    $display("D\t Q \t Qbar \t Reset");
    $monitor("%b\t %b\t %b\t %b", t, q, qbar, reset);
    t=0; reset=1; #3
    t=1; reset=1; #3
    t=0; reset=0; #5
    t=0; reset=1; #3
    t=1; reset=0; #3
    t=0; reset=1; #5
end

```

```
$finish;  
end  
initial begin  
  clk=0; reset=1;  
  for(i=0; i<50; i++) begin  
    #3 clk= ~clk;  
  end  
end  
endmodule
```



**9.(b) .Write a program in Verilog to implement JK Flip Flop using positive edge triggered clock. Verify logic using truth table and time wave**

```
module jkff(input [1:0] jk,input clk,output q,output qb);
    reg q,qb;
    always@(posedge clk)
    begin
        case(jk)
            2'b00:q=q;
            2'b01:q=0;
            2'b10:q=1;
            2'b11:q=~q;
        endcase
        qb=~q;
    end
endmodule

module test;
    reg [1:0]jk;
    reg clk,i;
    wire q,qb;
    jkff s(jk,clk,q,qb);
    initial
    begin
        $dumpfile("first.vcd");
        $dumpvars(1,test);
        $display("clk\tjk1\tjk0\tq\t~q");
        $monitor("%b\t%b\t%b\t%b\t%b",clk,jk[1],jk[0],q,qb);
        jk=2'b00;#10
        jk=2'b01;#10
        jk=2'b10;#10
        jk=2'b11;#10
        $finish;
    end
    initial
    begin
        clk=0;
        for(i=0;i<=20;i++)
            #5 clk=~clk;
        end
    endmodule
```

C:\ Command Prompt

Microsoft Windows [Version 10.0.17134.407]  
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd C:\iverilog\bin

C:\iverilog\bin>iverilog first.v

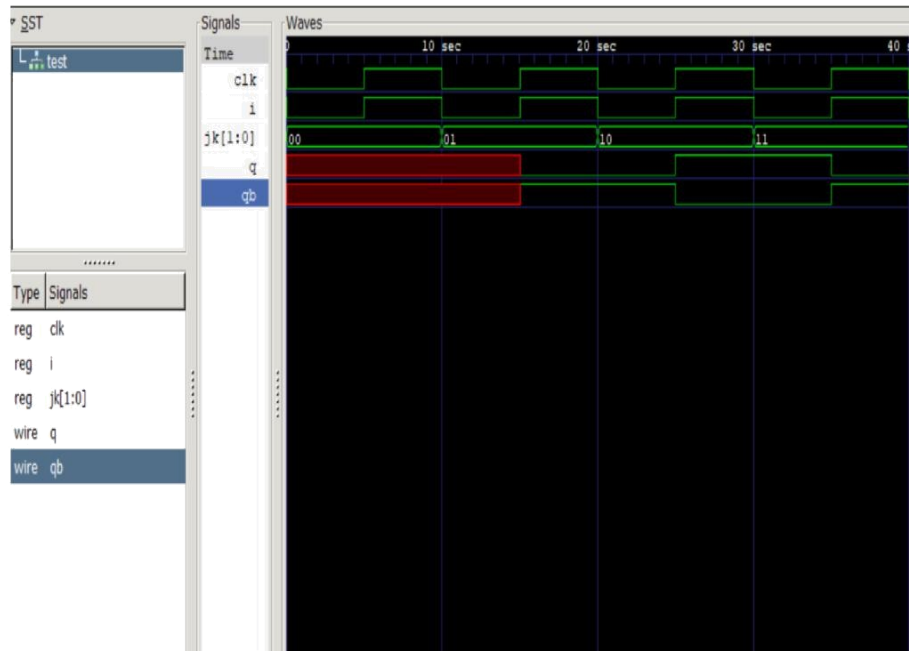
C:\iverilog\bin>vvp a.out

VCD info: dumpfile first.vcd opened for output.

clk	jk1	jk0	q	~q
0	0	0	x	x
1	0	0	x	x
0	0	1	x	x
1	0	1	0	1
0	1	0	0	1
1	1	0	1	0
0	1	1	1	0
1	1	1	0	1
0	1	1	0	1

C:\iverilog\bin>





**10. Write a program in Verilog for 4-bit counter using reset and enable. Verify the logic using truth table and time wave.**

**Ans:**

```
module counter1(input reset,input clk,input
enable,output [3:0]counter);
reg [3:0]counter;
always@(posedge clk)begin 1
if(reset==1)
counter=4'b0000;
else if(enable==1)
counter=counter+4'b0001;
end
endmodule

module test;
reg reset;
reg clk;
reg enable;
integer i;
wire [3:0]counter;
counter1 c(reset,clk,enable,counter);
initial begin
clk=0;
for( i=0;i<100;i++)
#2 clk=~clk;
$finish;
end
initial begin
$display("reset clk enable counter");
$monitor("%b\t%b\t%b\t%b",reset,clk,enable,counter);
```

```
$dumpfile("abc.vcd");
```

```
$dumpvars(0,test);
```

```
reset=1;
```

```
enable=0;
```

```
#10
```

```
reset=0;
```

```
enable=1;
```

```
#60
```

```
reset=0;
```

```
enable=0;
```

```
#10
```

```
reset=0;
```

```
enable=1;
```

```
#10
```

```
reset=1;
```

```
enable=1;
```

```
#10
```

```
$finish;
```

```
end
```

```
endmodule
```

Command Window output:-

0	0	xxxx
1	0	0000
1	0	0000
1	0	0000
1	0	0000
0	1	0001
0	0	0001
0	1	0010
0	0	0010
0	1	0011
0	0	0011
0	1	0100
0	0	0100
0	1	0101
0	0	0101
0	1	0110
0	0	0110
0	1	0111
0	0	0111
0	1	1000
0	0	1000
0	1	1001
0	0	1001
0	1	1010
0	0	1010
0	1	1011
0	0	1011
0	1	1100
0	0	1100
0	1	1101
0	0	1101
0	1	1110
0	0	1110
0	1	1111
0	0	1111

GTKWAVE:-

