B.M.S College of Engineering
(Autonomous Institution affiliated to VTU, Belagavi)
Bengaluru – 19

Department of Computer Science and Engineering

Report on
"Verilog Programs using Structural Modeling, Behavioral Modeling and Data Flow Modeling"

LOGIC DESIGN - 19CS3PCLOD

(Autonomous Scheme 2019)

Submitted by
Name: Khushil M Sindhwad
USN: 1BM19CS072

B.M.S. College of Engineering

(Autonomous Institution affiliated to VTU, Belagavi)

Bengaluru - 19

Department of Computer Science and Engineering

Certificate

This is to certify that Mr. **Khushil M Sindhwad** has satisfactorily completed the course of Experiments in **LOGIC DESIGN course** prescribed by the Department during the year **2020-2021.**

Name of the Candidate: Khushil M Sindhwad

USN No.: **1BM19CS072** Semester**: 3rd**

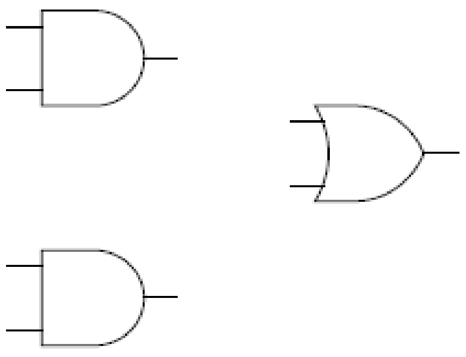
Marks	
Max. Marks	Obtained
10	

Marks in Words	

Signature of the staff in-charge **Head of the Department**

Date:

Verilog Program List
19CS3PCLOD

Serial No.	**Title**
	CYCLE I **Structural Modeling**
•	Write HDL implementation for the following Logic • AND/OR/NOT Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for the following Logic • NAND/NOR Simulate the same using structural model and depict the timing diagram for valid inputs.
•	 Write HDL implementation for the following Logic

	Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 4:1 Multiplexer. Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 2-to-4 decoder. Simulate the same using structural model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 4-to-2 encoder. Simulate the same using structural model and depict the timing diagram for valid inputs.
	CYCLE II **Behavior Modeling**
•	Write HDL implementation for a RS flip-flop using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a JK flip-flop using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 4-bit right shift register using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 3-bit up-counter using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
	CYCLE III **Dataflow Modeling**
•	Write HDL implementation for AND/OR/NOT gates using data flow model. Simulate the same using Dataflow model and depict the timing diagram for valid inputs.
•	Write HDL implementation for a 3-bit full adder using data flow model. Simulate the same using Dataflow model and depict the timing diagram for valid inputs.

STRUCTURAL MODELING
Experiment 1

- Write HDL implementation for the following Logic
    - AND/OR/NOT

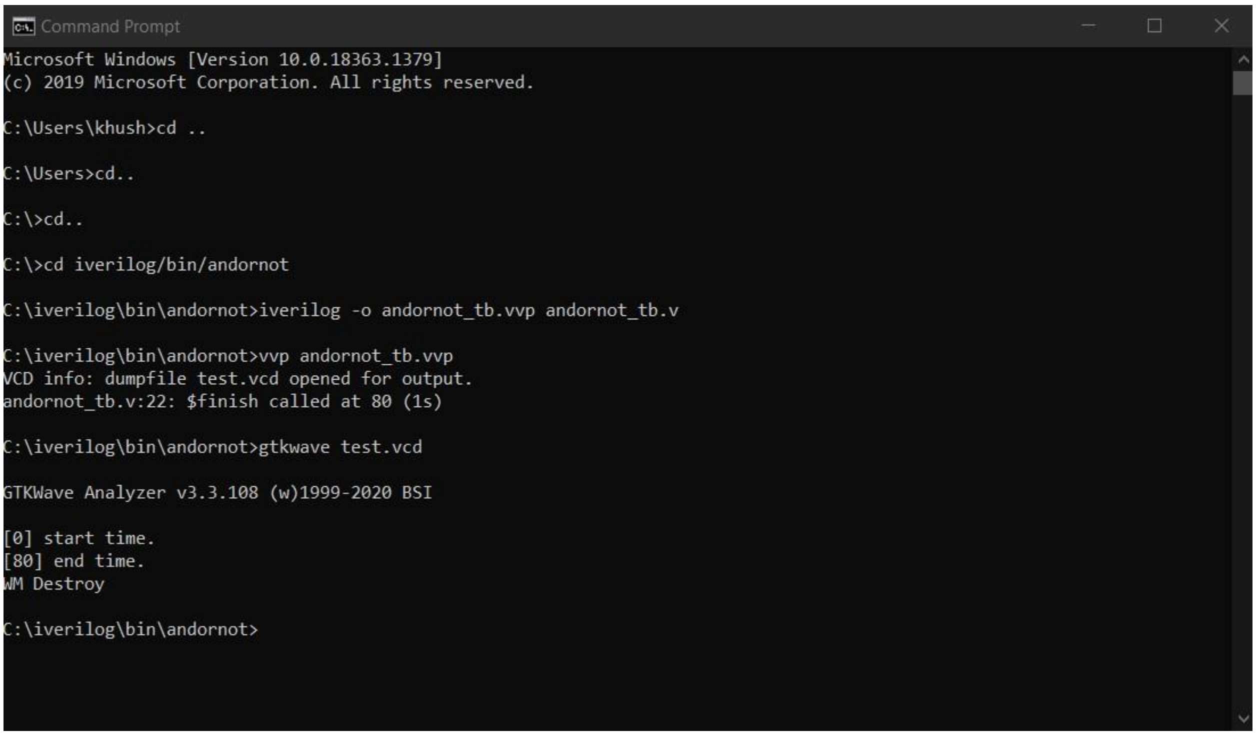
Simulate the same using structural model and depict the timing diagram for valid inputs.

MAIN MODULE (andornot.v)

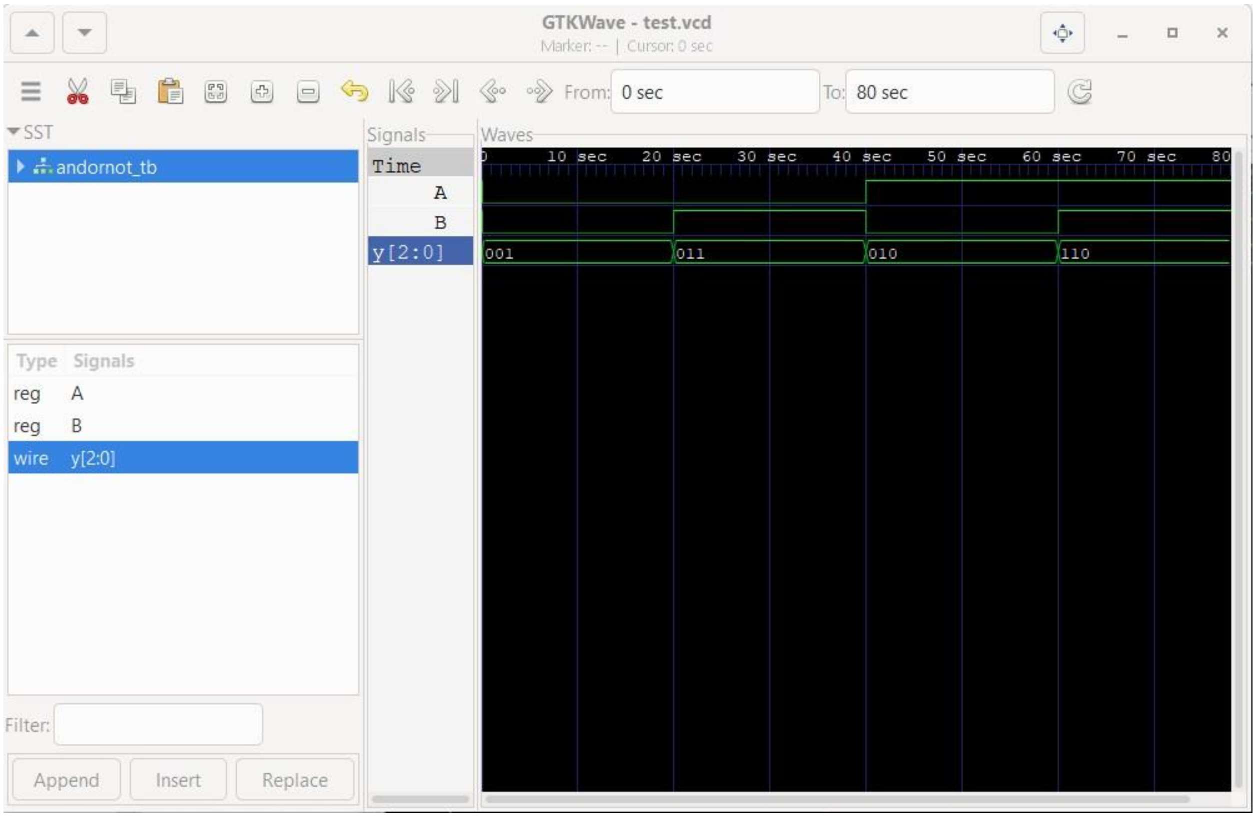
```
module andornot(a,b,y);
                    input a,b;
                    output [2:0]y;
                    and an(y[2],a,b);
                    or o(y[1],a,b);
                    not nt(y[0],a);
endmodule
```

TESTBENCH MODULE (andornot_tb.v)

```
`include "andornot.v"
module andornot_tb;
reg A,B;
wire [2:0]y;
andornot ob(A,B,y);
initial
begin
$dumpfile("test.vcd");
$dumpvars(0,andornot_tb);
                    A=1'b0;
  B=1'b0;
  #20
  A=1'b0;
  B=1'b1;
  #20
  A=1'b1;
  B=1'b0;
  #20
  A=1'b1;
  B=1'b1;
  #20
  $finish;
end
endmodule
```



```
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\khush>cd ..

C:\Users>cd..

C:\>cd..

C:\>cd iverilog/bin/andornot

C:\iverilog\bin\andornot>iverilog -o andornot_tb.vvp andornot_tb.v

C:\iverilog\bin\andornot>vvp andornot_tb.vvp
VCD info: dumpfile test.vcd opened for output.
andornot_tb.v:22: $finish called at 80 (1s)

C:\iverilog\bin\andornot>gtkwave test.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[80] end time.
WM Destroy

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



Type	Signals
reg	A
reg	B
wire	y[2:0]

Experiment 2

Write HDL implementation for the following Logic

- NAND/NOR

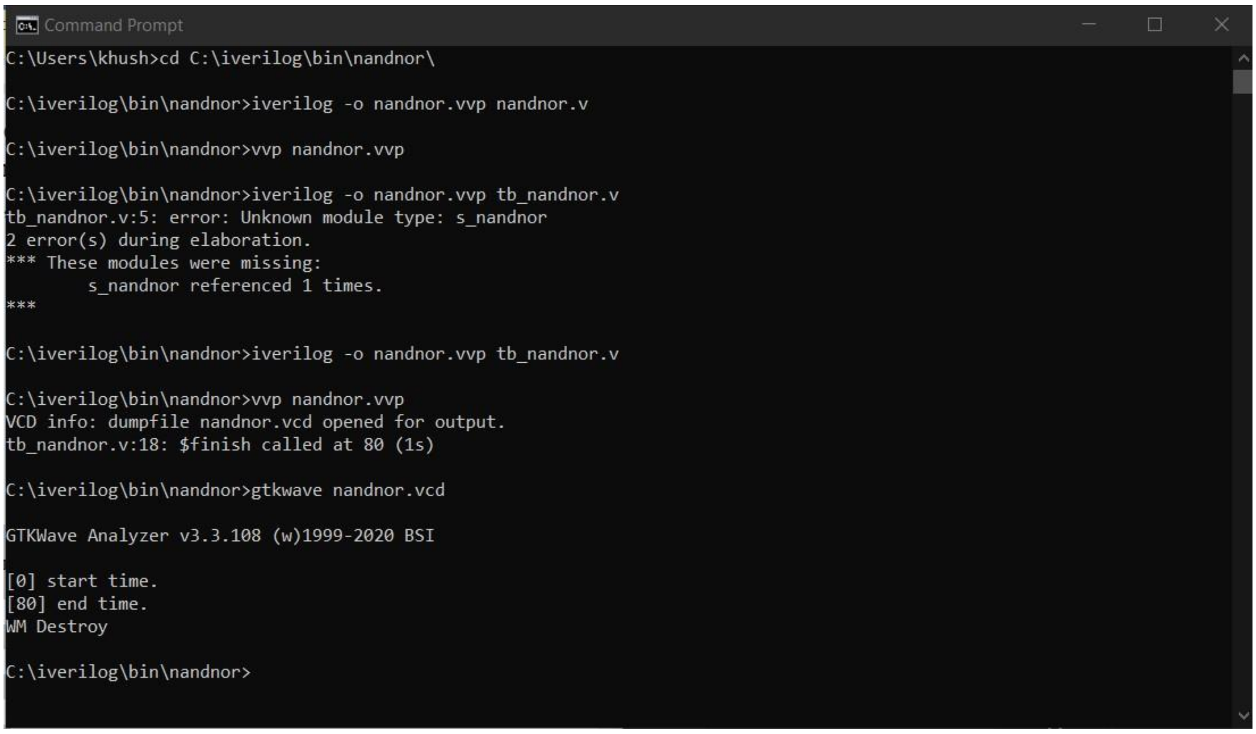
Simulate the same using structural model and depict the timing diagram for valid inputs.

MAIN MODULE (nandnor.v)

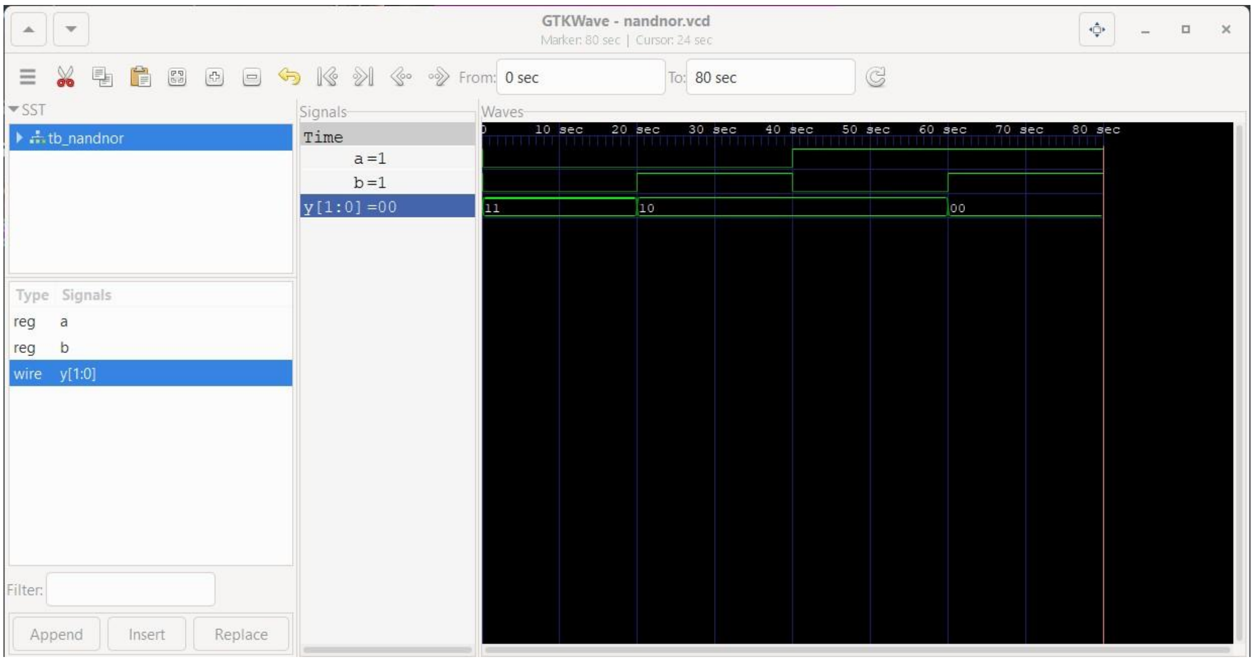
```
module nandnor(a,b,y);
        input a,b;
        output [1:0]y;
        nand na(y[1],a,b);
        nor no(y[0],a,b);
endmodule
```

TEST MODULE (tb_nandnor.v)

```
`include "nandnor.v"
module tb_nandnor;
        reg a,b;
        wire [1:0]y;
        nandnor ob(a,b,y);
        initial
        begin
        $dumpfile("nandnor.vcd");
        $dumpvars(0,tb_nandnor);
                a=1'b0;b=1'b0;
                #20
                a=1'b0;b=1'b1;
                #20
                a=1'b1;b=1'b0;
                #20
                a=1'b1;b=1'b1;
                #20
                $finish;
        end
endmodule
```

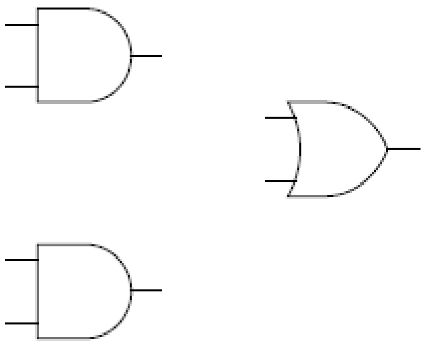


```
C:\Users\khush>cd C:\iverilog\bin\nandnor\

C:\iverilog\bin\nandnor>iverilog -o nandnor.vvp nandnor.v

C:\iverilog\bin\nandnor>vvp nandnor.vvp

C:\iverilog\bin\nandnor>iverilog -o nandnor.vvp tb_nandnor.v
tb_nandnor.v:5: error: Unknown module type: s_nandnor
2 error(s) during elaboration.
*** These modules were missing:
        s_nandnor referenced 1 times.
***

C:\iverilog\bin\nandnor>iverilog -o nandnor.vvp tb_nandnor.v

C:\iverilog\bin\nandnor>vvp nandnor.vvp
VCD info: dumpfile nandnor.vcd opened for output.
tb_nandnor.v:18: $finish called at 80 (1s)

C:\iverilog\bin\nandnor>gtkwave nandnor.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[80] end time.
WM Destroy

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



Experiment 3

Write HDL implementation for the following Logic



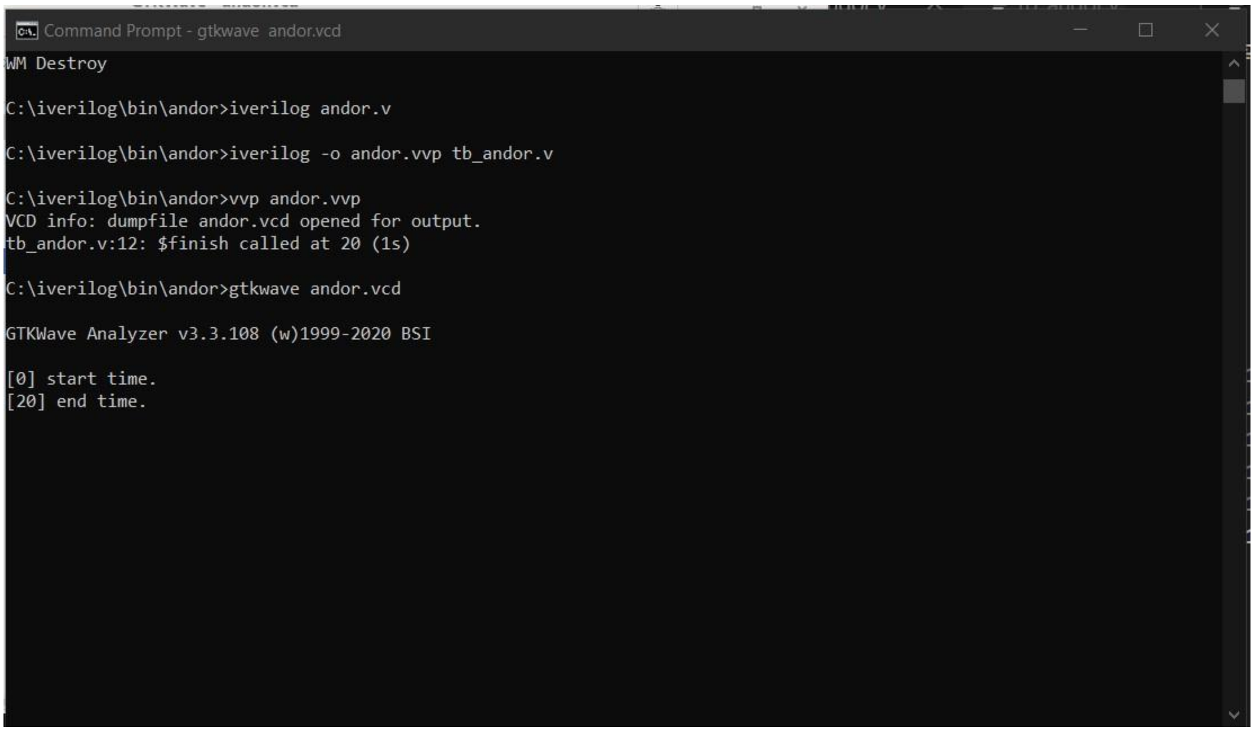
Simulate the same using structural model and depict the timing diagram for valid inputs.

MAIN MODULE (andor.v)

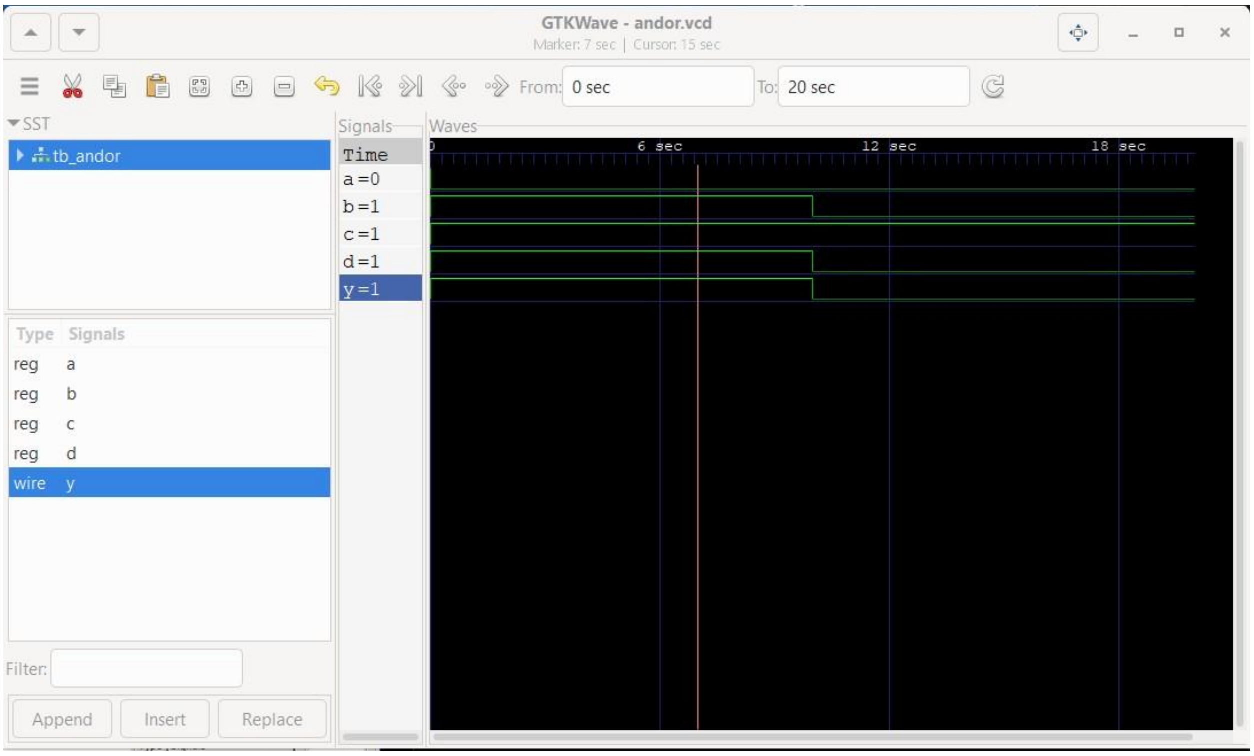
```
module andor(A,B,C,D,Y);
input A,B,C,D;
output Y;
wire al,a2;
and gl(al,A,B);
and g2(a2,C,D);
or g3(Y,al,a2);
endmodule
```

TEST MODULE (tb_andor.v)

```
`include "andor.v"
module tb_andor;
reg a,b,c,d;
wire y;
andor ao(a,b,c,d,y);
initial
begin
$dumpfile("andor.vcd");
$dumpvars(0,tb_andor);
a=0; b=1; c=1; d=1; #10
a=0; b=0; c=1; d=0; #10
$finish;
end
endmodule
```



```
Command Prompt - gtkwave  andor.vcd
WM Destroy

C:\iverilog\bin\andor>iverilog andor.v

C:\iverilog\bin\andor>iverilog -o andor.vvp tb_andor.v

C:\iverilog\bin\andor>vvp andor.vvp
VCD info: dumpfile andor.vcd opened for output.
tb_andor.v:12: $finish called at 20 (1s)

C:\iverilog\bin\andor>gtkwave andor.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

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



GTKWave - andor.vcd
Marker: 7 sec | Cursor: 15 sec

From: 0 sec To: 20 sec

SST
tb_andor

Type	Signals
reg	a
reg	b
reg	c
reg	d
wire	y

```
Time
a =0
b =1
c =1
d =1
y =1
```

Experiment 4

Write HDL implementation for a 4:1 Multiplexer. Simulate the same using structural model and depict the timing diagram for valid inputs.

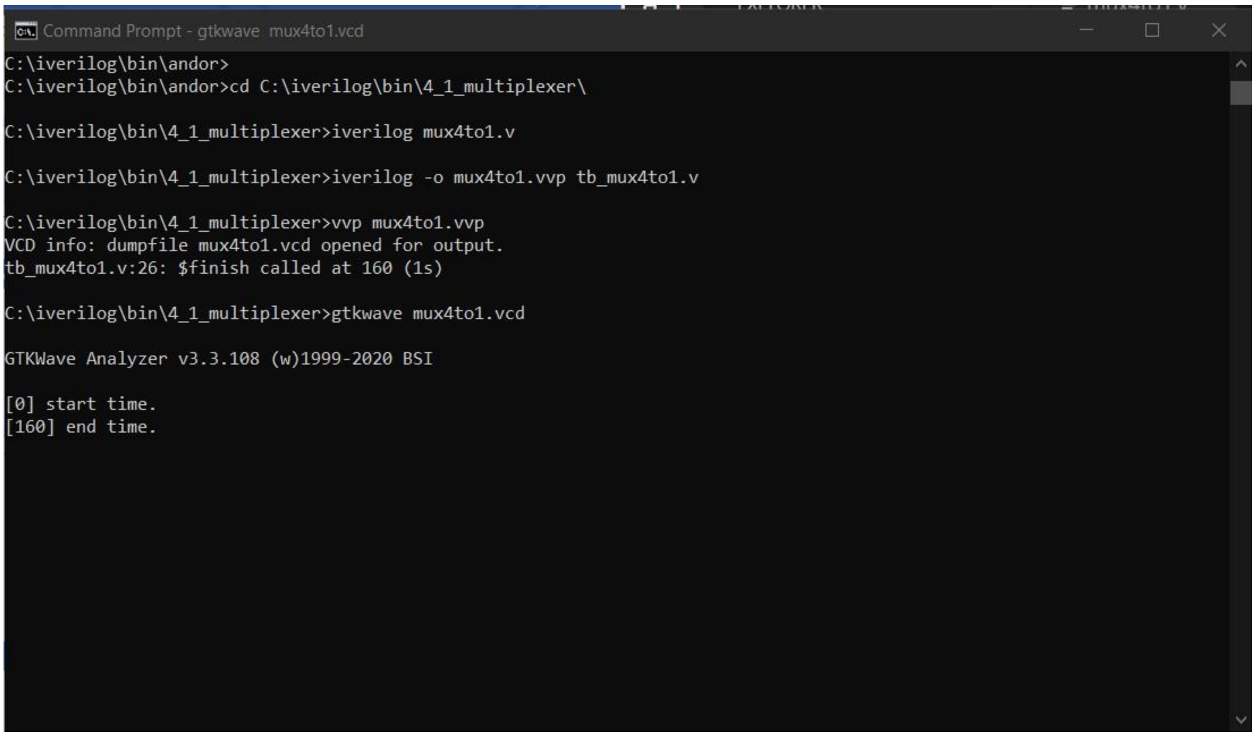
MAIN MODULE (mux4to1.v)

```
module mux4to1(a, b, c, d, sel1, sel2, y);
   input a, b, c, d, sel1, sel2;
   output y;
   wire and1, and2, and3, and4;
   and g1(and1, a, ~sel1, ~sel2);
   and g2(and2, b, ~sel1, sel2);
   and g3(and3, c, sel1, ~sel2);
   and g4(and4, d, sel1, sel2);
   or g5(y, and1, and2, and3, and4);

endmodule
```

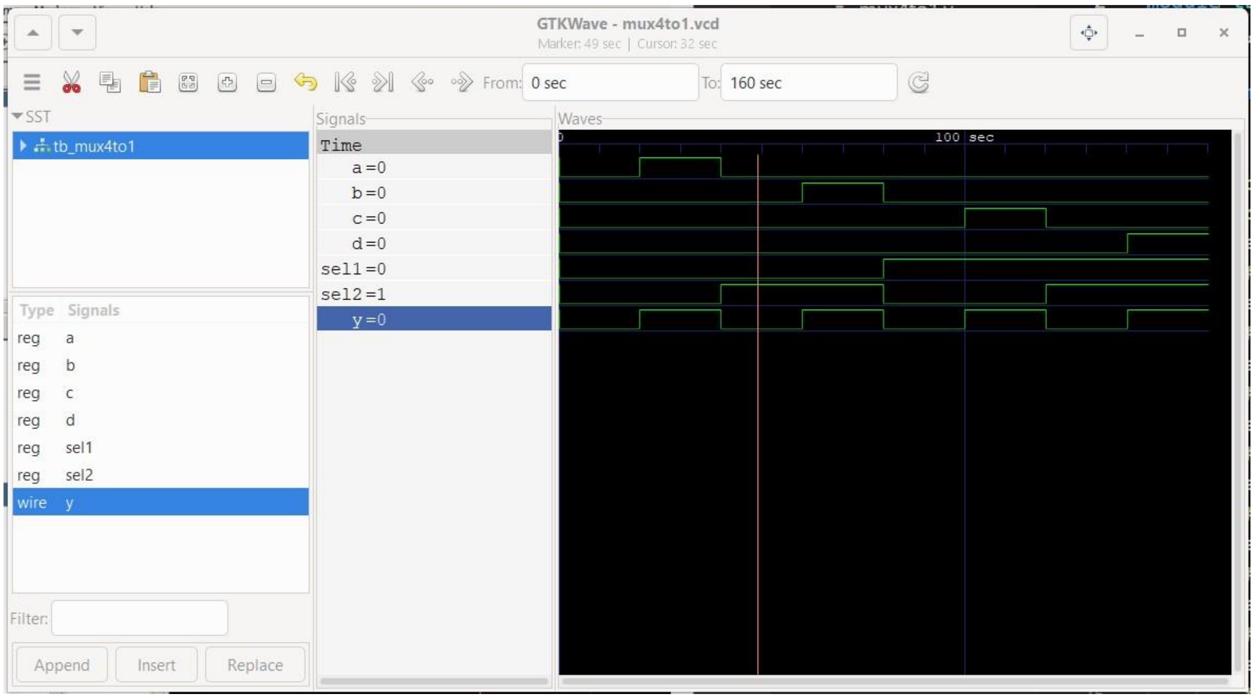
TESTBENCH MODULE (tb_mux4to1.v)

```
`include "mux4to1.v"
module tb_mux4to1;
reg a, b, c, d, sel1, sel2;
wire y;
mux4to1 mg(a, b, c, d, sel1, sel2, y);
initial
begin
      $dumpfile("mux4to1.vcd");
                              $dumpvars(0, tb_mux4to1);
                               a = 0; b = 0; c = 0; d = 0; sel1 = 0; sel2 = 0;
                               #20
                               a = 1; b = 0; c = 0; d = 0; sel1 = 0; sel2 = 0;
                               #20
                               a = 0; b = 0; c = 0; d = 0; sel1 = 0; sel2 = 1;
                               #20
                               a = 0; b = 1; c = 0; d = 0; sel1 = 0; sel2 = 1;
                               #20
```

```
                    a = 0; b = 0; c = 0; d = 0; sel1 = 1; sel2 = 0;
                    #20
                    a = 0; b = 0; c = 1; d = 0; sel1 = 1; sel2 = 0;
                    #20
                    a = 0; b = 0; c = 0; d = 0; sel1 = 1; sel2 = 1;
                    #20
                    a = 0; b = 0; c = 0; d = 1; sel1 = 1; sel2 = 1;
                    #20
                    $finish;
                   end
endmodule
```



```
C:\iverilog\bin\andor>
C:\iverilog\bin\andor>cd C:\iverilog\bin\4_1_multiplexer\

C:\iverilog\bin\4_1_multiplexer>iverilog mux4to1.v

C:\iverilog\bin\4_1_multiplexer>iverilog -o mux4to1.vvp tb_mux4to1.v

C:\iverilog\bin\4_1_multiplexer>vvp mux4to1.vvp
VCD info: dumpfile mux4to1.vcd opened for output.
tb_mux4to1.v:26: $finish called at 160 (1s)

C:\iverilog\bin\4_1_multiplexer>gtkwave mux4to1.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

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



Experiment 5

Write HDL implementation for a 2-to-4 decoder. Simulate the same using structural model and depict the timing diagram for valid inputs.

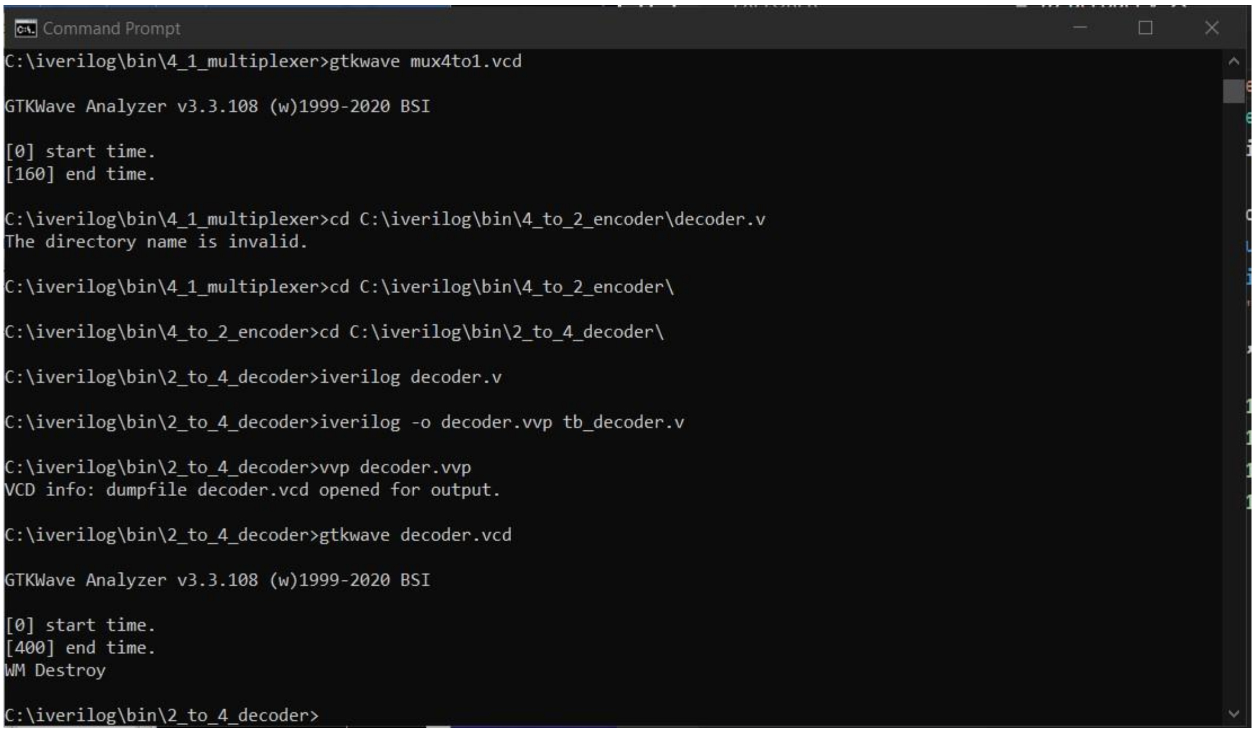
MAIN MODULE (decoder.v)

```
module decoder(Do,Din,En);
    input En;
    input[1:0]Din;
    output [3:0]Do;
    reg [3:0]Do;
    always@(En or Din)
    begin
        if(En)
        begin
            case(Din)
                2'b00:Do=4'b0001;
                2'b01:Do=4'b0010;
                2'b10:Do=4'b0100;
                2'b11:Do=4'b1000;
                default:Do=4'bzzzz;
            endcase
        end
    end
endmodule
```

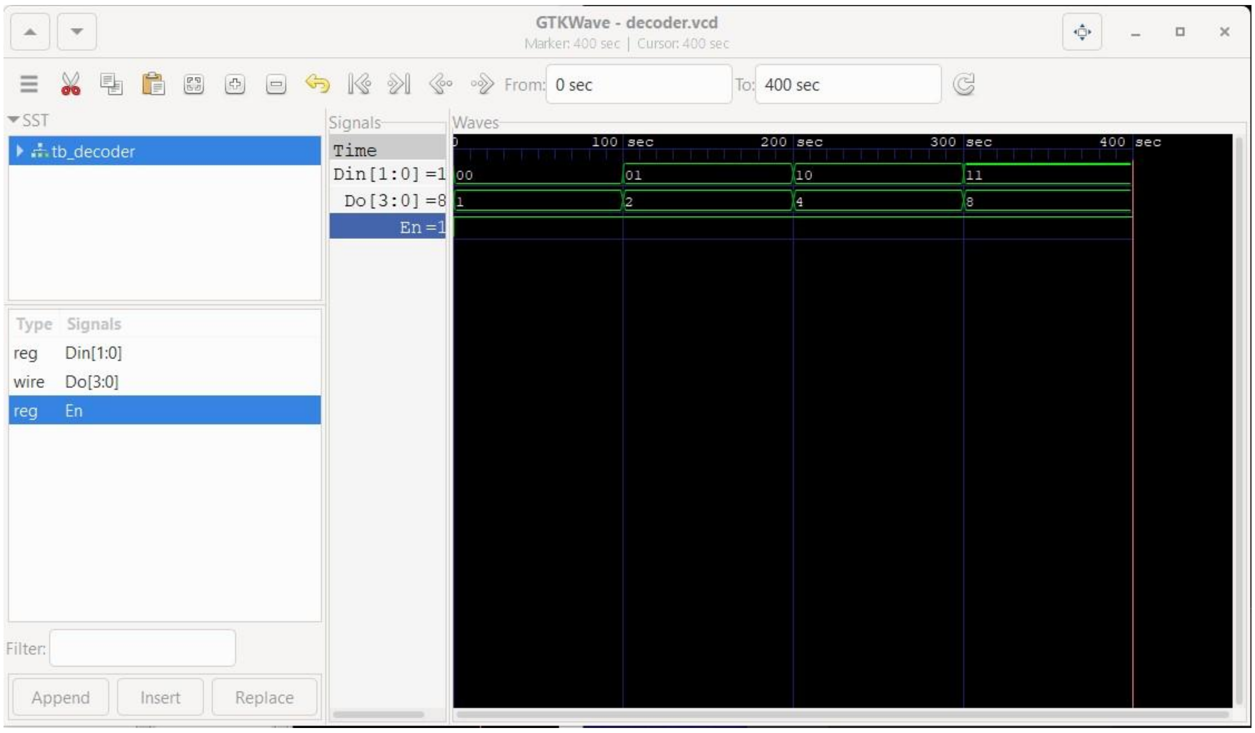
TEST BENCH MODULE (tb_decoder.v)

```
`include "decoder.v"
module tb_decoder;
 reg [1:0]Din;
 reg En;
```

```
wire [3:0]Do;

decoder  udue(.Do(Do),.Din(Din),.En(En));

initial begin

 $dumpfile("decoder.vcd");

$dumpvars(0,tb_decoder);

En=1;

Din=2'b00;#100;

Din=2'b01;#100;

Din=2'b10;#100;

Din=2'b11;#100;

end

endmodule
```



```
C:\iverilog\bin\4_1_multiplexer>gtkwave mux4to1.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[160] end time.

C:\iverilog\bin\4_1_multiplexer>cd C:\iverilog\bin\4_to_2_encoder\decoder.v
The directory name is invalid.

C:\iverilog\bin\4_1_multiplexer>cd C:\iverilog\bin\4_to_2_encoder\

C:\iverilog\bin\4_to_2_encoder>cd C:\iverilog\bin\2_to_4_decoder\

C:\iverilog\bin\2_to_4_decoder>iverilog decoder.v

C:\iverilog\bin\2_to_4_decoder>iverilog -o decoder.vvp tb_decoder.v

C:\iverilog\bin\2_to_4_decoder>vvp decoder.vvp
VCD info: dumpfile decoder.vcd opened for output.

C:\iverilog\bin\2_to_4_decoder>gtkwave decoder.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[400] end time.
WM Destroy

C:\iverilog\bin\2_to_4_decoder>
```



Experiment 6

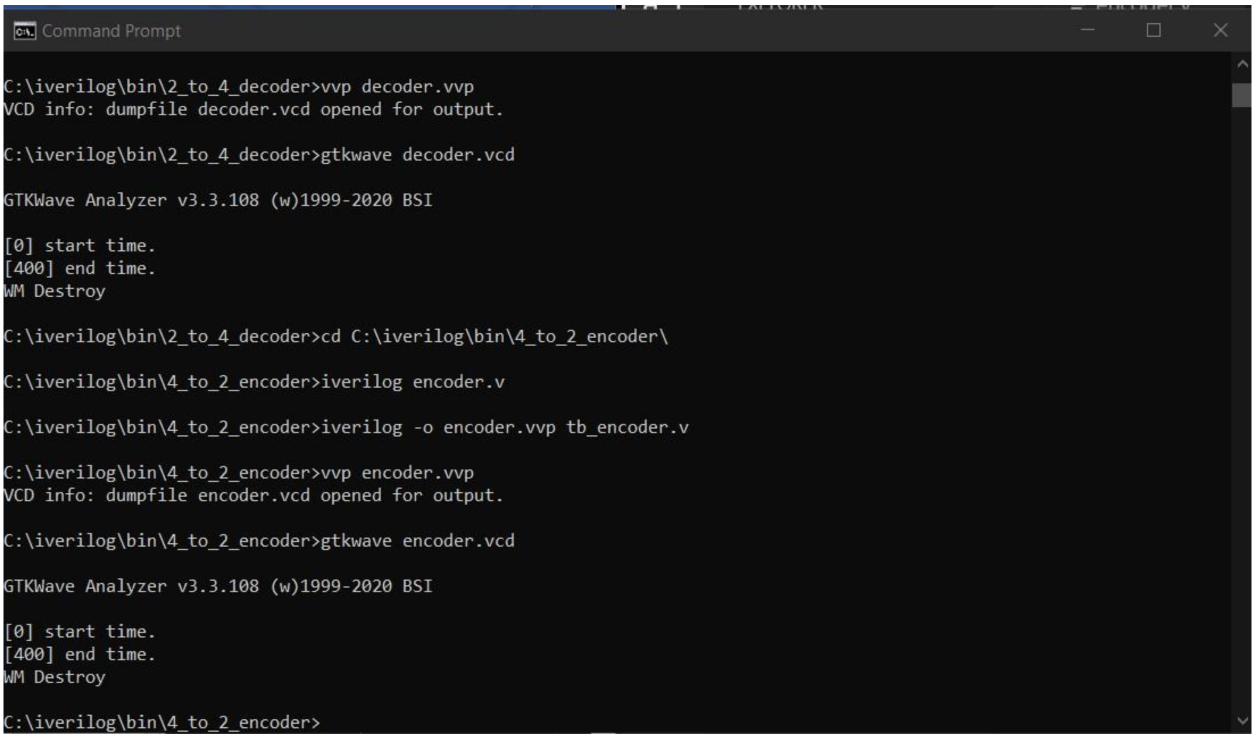
Write HDL implementation for a 4-to-2 encoder. Simulate the same using structural model and depict the timing diagram for valid inputs.

MAIN MODULE (encoder.v)

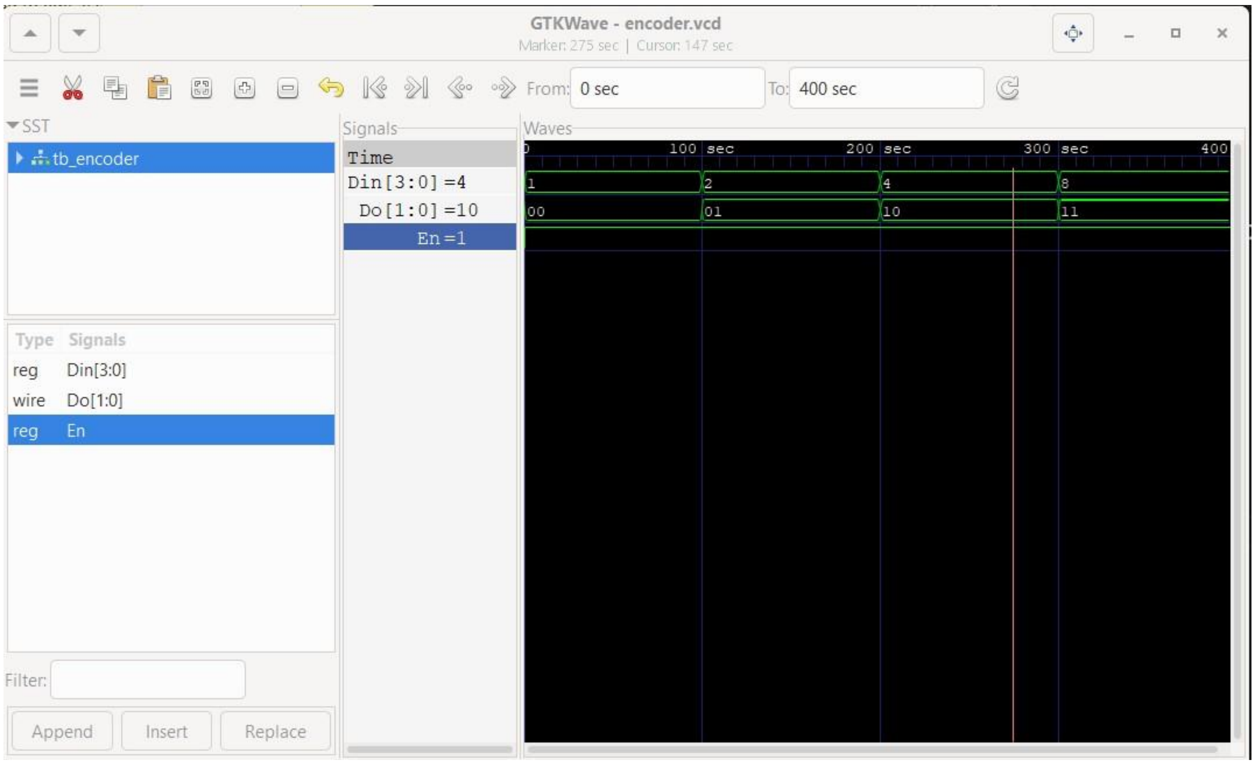
```
module encoder(Do,Din,En);
input En;
input[3:0]Din;

output [1:0]Do;
reg [1:0]Do;
always@(En or Din)
begin
if(En)
begin
case(Din)
        4'b0001:Do=2'b00;
        4'b0010:Do=2'b01;
        4'b0100:Do=2'b10;
        4'b1000:Do=2'b11;
default:Do=2'bzz;
endcase
end
end
endmodule
```

TESTBENCH MODULE (tb_encoder.v)

```
`include "encoder.v"
module tb_encoder;
 reg [3:0]Din;
 reg En;
wire [1:0]Do;
encoder  mux1(.Do(Do),.Din(Din),.En(En));
initial begin
 $dumpfile("encoder.vcd");
$dumpvars(0,tb_encoder);
En=1;
Din=4'b0001;#100;
Din=4'b0010;#100;
Din=4'b0100;#100;
Din=4'b1000;#100;
end
endmodule
```



```
C:\iverilog\bin\2_to_4_decoder>vvp decoder.vvp
VCD info: dumpfile decoder.vcd opened for output.

C:\iverilog\bin\2_to_4_decoder>gtkwave decoder.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[400] end time.
WM Destroy

C:\iverilog\bin\2_to_4_decoder>cd C:\iverilog\bin\4_to_2_encoder\

C:\iverilog\bin\4_to_2_encoder>iverilog encoder.v

C:\iverilog\bin\4_to_2_encoder>iverilog -o encoder.vvp tb_encoder.v

C:\iverilog\bin\4_to_2_encoder>vvp encoder.vvp
VCD info: dumpfile encoder.vcd opened for output.

C:\iverilog\bin\4_to_2_encoder>gtkwave encoder.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[400] end time.
WM Destroy

C:\iverilog\bin\4_to_2_encoder>
```



BEHAVIOR MODELING

Experiment 7

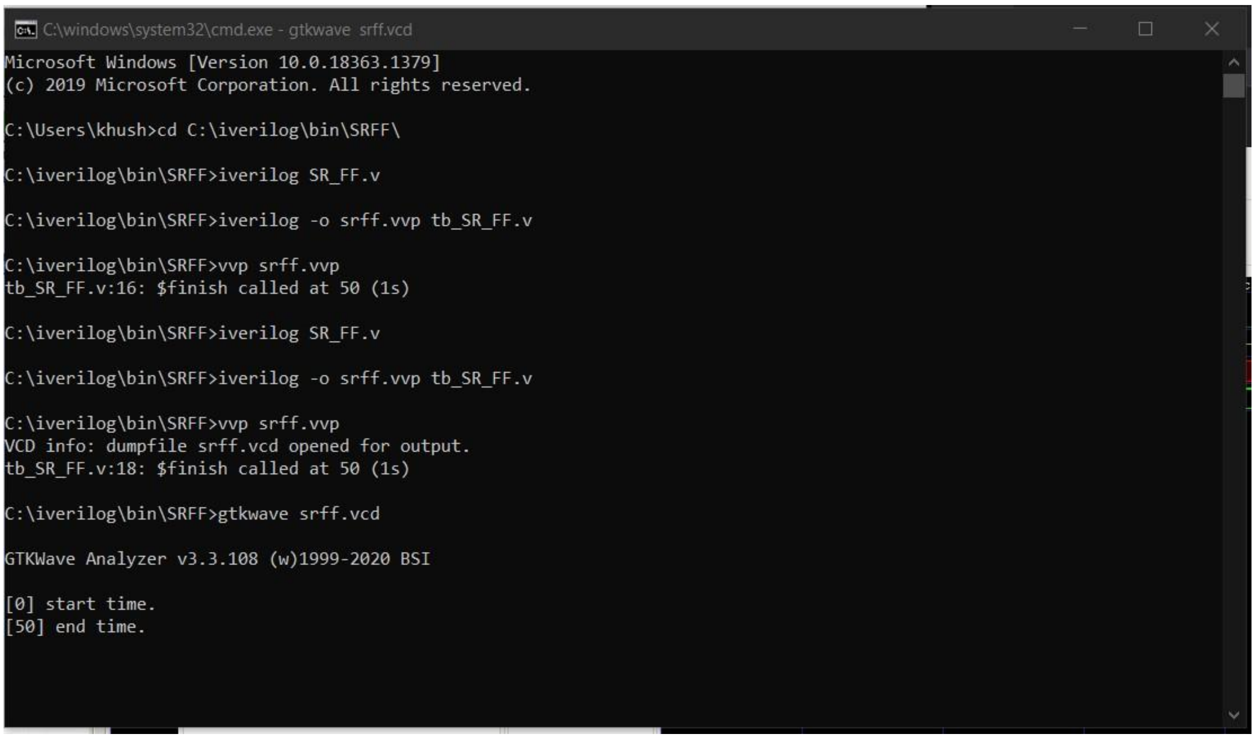
Write HDL implementation for a SR flip-flop using behavioral model. Simulate the same using behavioral model and depict the timing diagram for valid inputs.

MAIN MODULE (SR_FF.v)

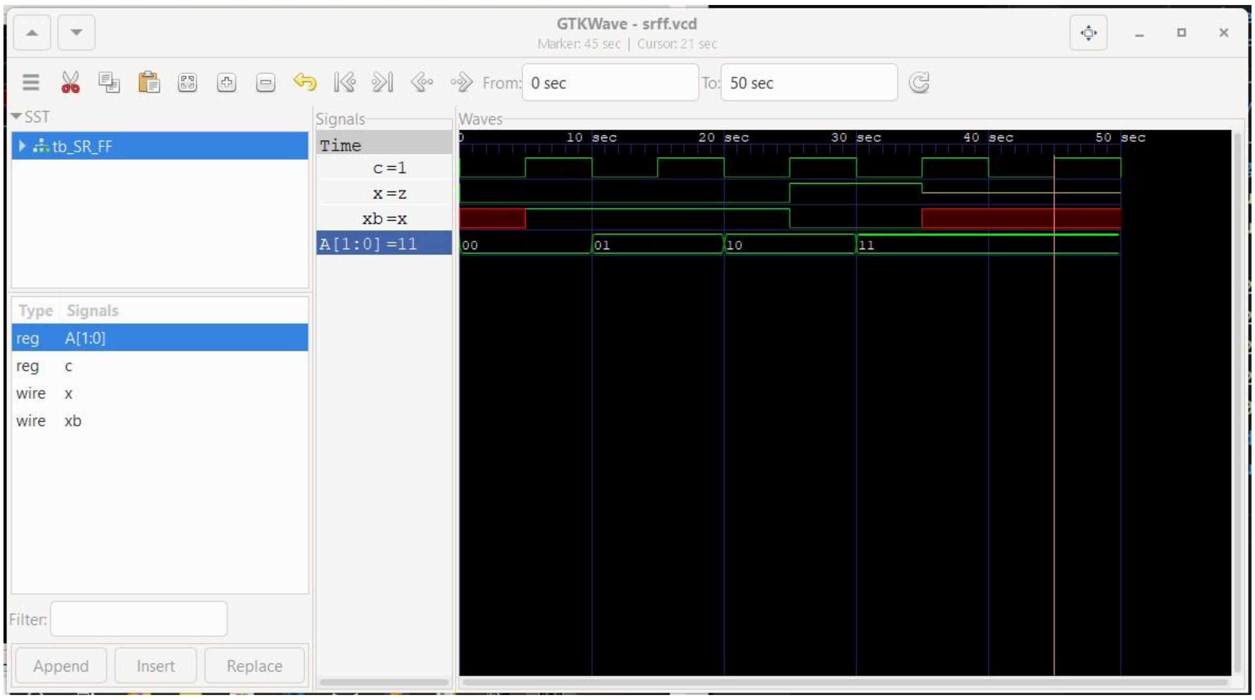
```
module SR_FF (sr, clk, q, qb);
input [1:0] sr;
input clk;
output reg q=1'b0;
output reg qb;
always @ (posedge clk)
begin
      case (sr)
                  2'b00 : q = q ;
                        2'b01 : q = 1'b0 ;
                        2'b10 : q = 1'b1 ;
                        2'b11 : q = 1'bz ;
      endcase
                  qb =~ q;
      end
endmodule
```

TEST MODULE (tb_SR_FF.v)

```
`include "SR_FF.v"
module tb_SR_FF;
   reg [1:0] A;
   reg c;
   wire x, xb;
  SR_FF srff(A,c,x,xb);
  initial c=1'b0;
  always #5 c=~c;
  initial
    begin
    $dumpfile("srff.vcd");
    $dumpvars(0,tb_SR_FF);

    A=2'b00; #10
    A=2'b01;#10
    A=2'b10;#10
    A=2'b11;
    #20 $finish;
    end
endmodule
```



```
C:\windows\system32\cmd.exe - gtkwave  srff.vcd
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\khush>cd C:\iverilog\bin\SRFF\

C:\iverilog\bin\SRFF>iverilog SR_FF.v

C:\iverilog\bin\SRFF>iverilog -o srff.vvp tb_SR_FF.v

C:\iverilog\bin\SRFF>vvp srff.vvp
tb_SR_FF.v:16: $finish called at 50 (1s)

C:\iverilog\bin\SRFF>iverilog SR_FF.v

C:\iverilog\bin\SRFF>iverilog -o srff.vvp tb_SR_FF.v

C:\iverilog\bin\SRFF>vvp srff.vvp
VCD info: dumpfile srff.vcd opened for output.
tb_SR_FF.v:18: $finish called at 50 (1s)

C:\iverilog\bin\SRFF>gtkwave srff.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

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



Experiment 8

Write HDL implementation for a JK flip-flop using behavioral model. Simulate the same using behavioral model and depict the timing diagram for valid inputs.

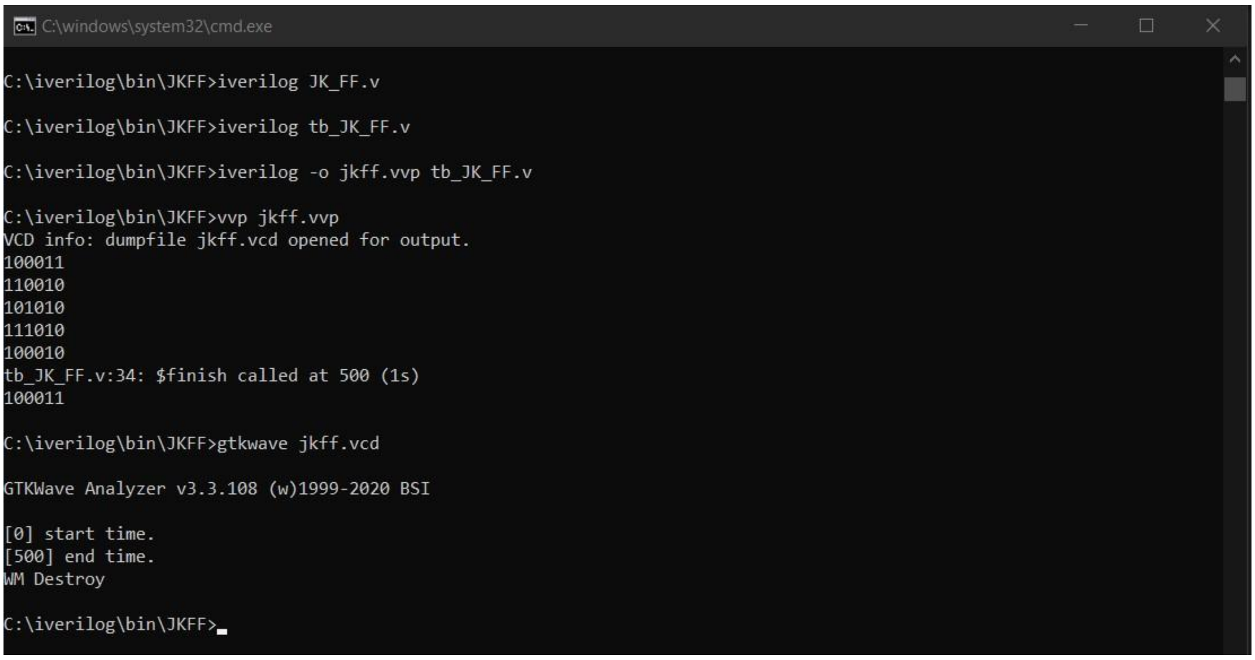
MAIN MODULE (JK_FF.v)

```
module JK_FF(j,k,clk,reset,q,q_bar);
input j,k,clk,reset;
output q,q_bar;
wire j,k,clk,reset;
reg q,q_bar;

always@(posedge clk)begin

if(reset)begin
q=1'b0;
q_bar=1'b1;

end else begin

case({j,k})
{1'b0,1'b0}:begin q=q;q_bar=q_bar;end
{1'b0,1'b1}:begin q=1'b0;q_bar=1'b1;end
{1'b1,1'b0}:begin q=1'b1;q_bar=1'b0;end
{1'b1,1'b1}:begin q=~q;q_bar=~q_bar;end
endcase
end
end
endmodule
```

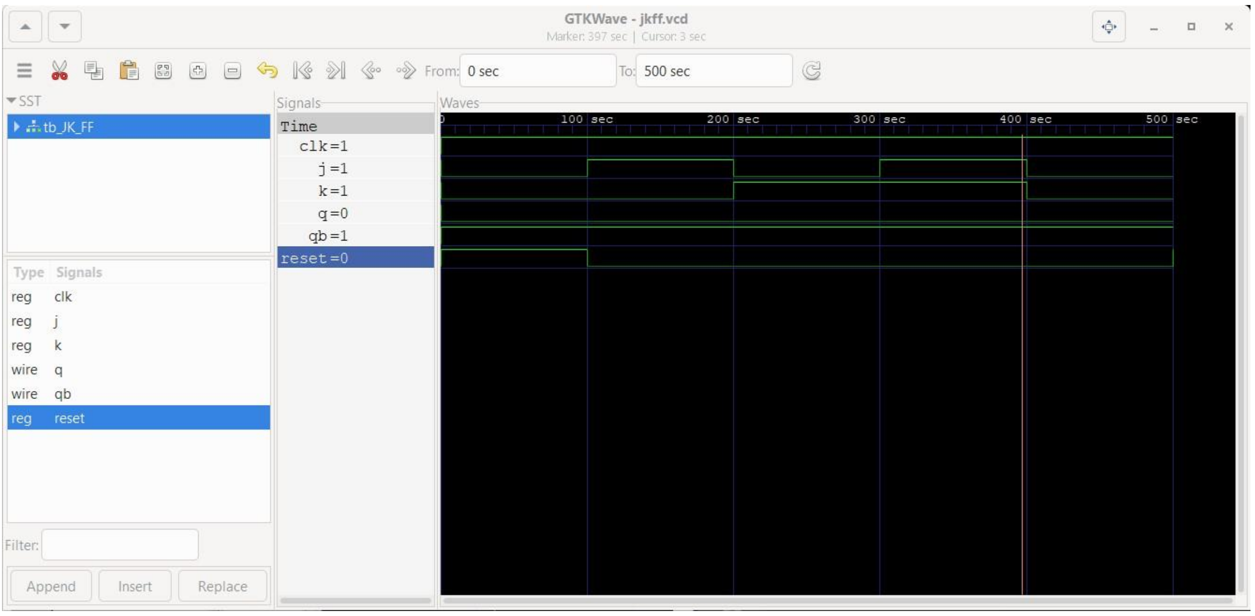
TEST MODULE (tb_JK_FF.v)

```
`include "JK_FF.v"
module tb_JK_FF;
reg clk;
reg reset;
reg j,k;
wire q;
wire qb;
JK_FF flipflop(.clk(clk),.reset(reset),.j(j),.k(k),.q(q),.q_bar(qb));
initial begin
$dumpfile("jkff.vcd");
$dumpvars(0,tb_JK_FF);
$monitor(clk,j,k,q,qb,reset);
j=1'b0;
k=1'b0;
reset=1;
clk=1;
#100
reset=0;
j=1'b1;
k=1'b0;
```

```
#100
reset=0;
j=1'b0;
k=1'b1;
#100
reset=0;
j=1'b1;
k=1'b1;
#100
reset=0;
j=1'b0;
k=1'b0;
#100
reset=1;$finish;
 end
 endmodule
```



```
C:\iverilog\bin\JKFF>iverilog JK_FF.v

C:\iverilog\bin\JKFF>iverilog tb_JK_FF.v

C:\iverilog\bin\JKFF>iverilog -o jkff.vvp tb_JK_FF.v

C:\iverilog\bin\JKFF>vvp jkff.vvp
VCD info: dumpfile jkff.vcd opened for output.
100011
110010
101010
111010
100010
tb_JK_FF.v:34: $finish called at 500 (1s)
100011

C:\iverilog\bin\JKFF>gtkwave jkff.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[500] end time.
WM Destroy

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



GTKWave - jkff.vcd
Marker: 397 sec | Cursor: 3 sec
From: 0 sec To: 500 sec

```
Time
clk=1
j=1
k=1
q=0
qb=1
reset=0
```

Type	Signals
reg	clk
reg	j
reg	k
wire	q
wire	qb
reg	reset

Experiment 9

Write HDL implementation for a 4-bit right shift register using behavioral model. Simulate the same using behavioral model and depict the timing diagram for valid inputs.

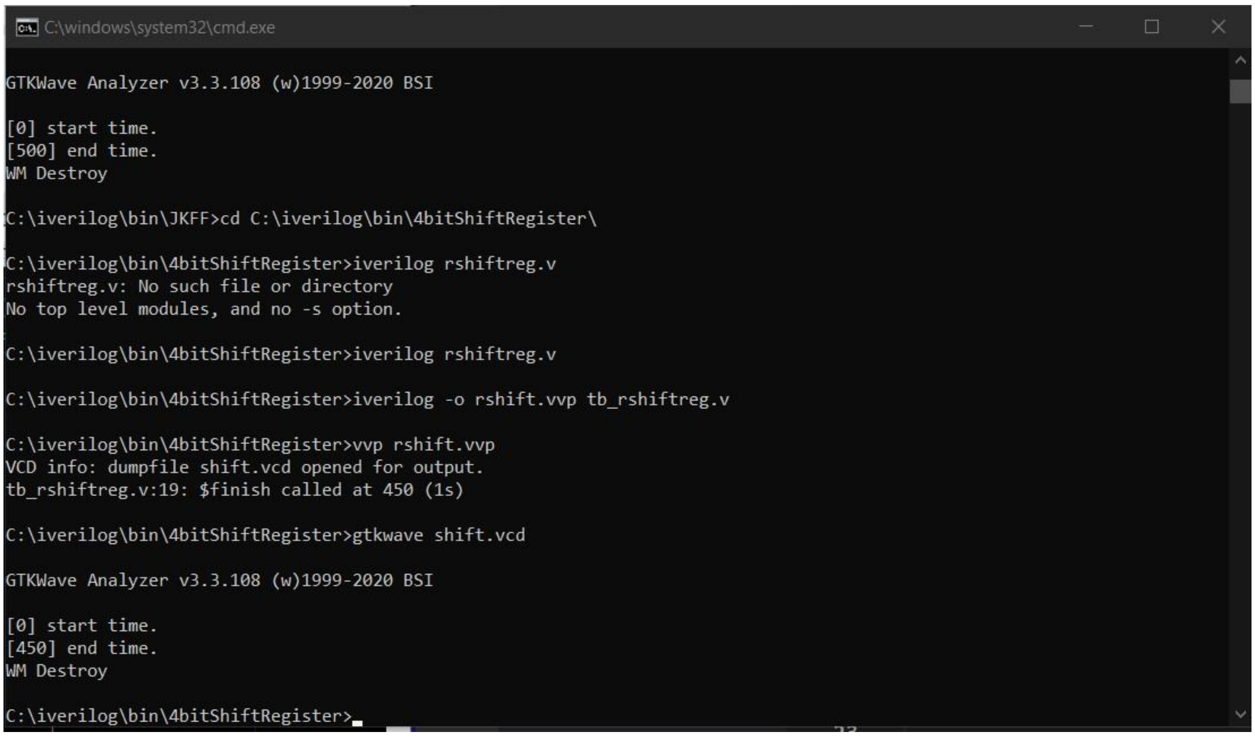
MAIN MODULE (rshiftreg.v)

```
module rshiftreg(input clk,input clrb,input sdr,output reg [3:0]q);
always @(posedge(clk),negedge(clrb))
if(~clrb)
q<=4'b0000;
else
q<={sdr,q[3:1]};
endmodule
```

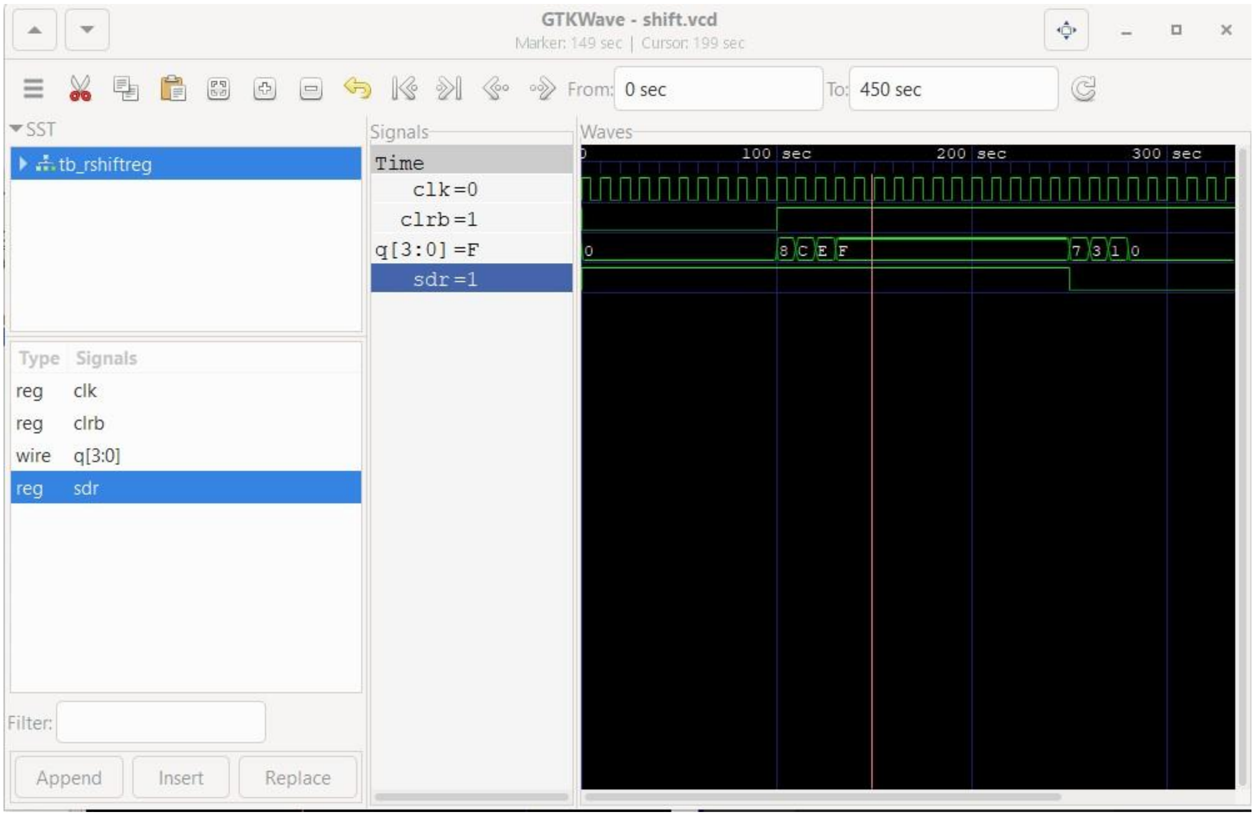
TEST MODULE (tb_rshiftreg.v)

```
`include "rshiftreg.v"

module tb_rshiftreg;

reg clk,clrb,sdr;

wire [3:0]q;

rshiftreg rs(clk,clrb,sdr,q);

initial

begin

$dumpfile("shift.vcd");

$dumpvars(0,tb_rshiftreg);

clk=1;

clrb=0;

sdr=1;

#100

clrb=1;

sdr=1;

#150

sdr=0;

#200
```

```
$finish;

end

always #5 clk=~clk;

endmodule
```



```
C:\windows\system32\cmd.exe

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[500] end time.
WM Destroy

C:\iverilog\bin\JKFF>cd C:\iverilog\bin\4bitShiftRegister\

C:\iverilog\bin\4bitShiftRegister>iverilog rshiftreg.v
rshiftreg.v: No such file or directory
No top level modules, and no -s option.

C:\iverilog\bin\4bitShiftRegister>iverilog rshiftreg.v

C:\iverilog\bin\4bitShiftRegister>iverilog -o rshift.vvp tb_rshiftreg.v

C:\iverilog\bin\4bitShiftRegister>vvp rshift.vvp
VCD info: dumpfile shift.vcd opened for output.
tb_rshiftreg.v:19: $finish called at 450 (1s)

C:\iverilog\bin\4bitShiftRegister>gtkwave shift.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[450] end time.
WM Destroy

C:\iverilog\bin\4bitShiftRegister>_
```



GTKWave - shift.vcd
Marker: 149 sec | Cursor: 199 sec

From: 0 sec To: 450 sec

SST
tb_rshiftreg

Type	Signals
reg	clk
reg	clrb
wire	q[3:0]
reg	sdr

Signals
Time
clk=0
clrb=1
q[3:0] =F
sdr=1

Experiment 10

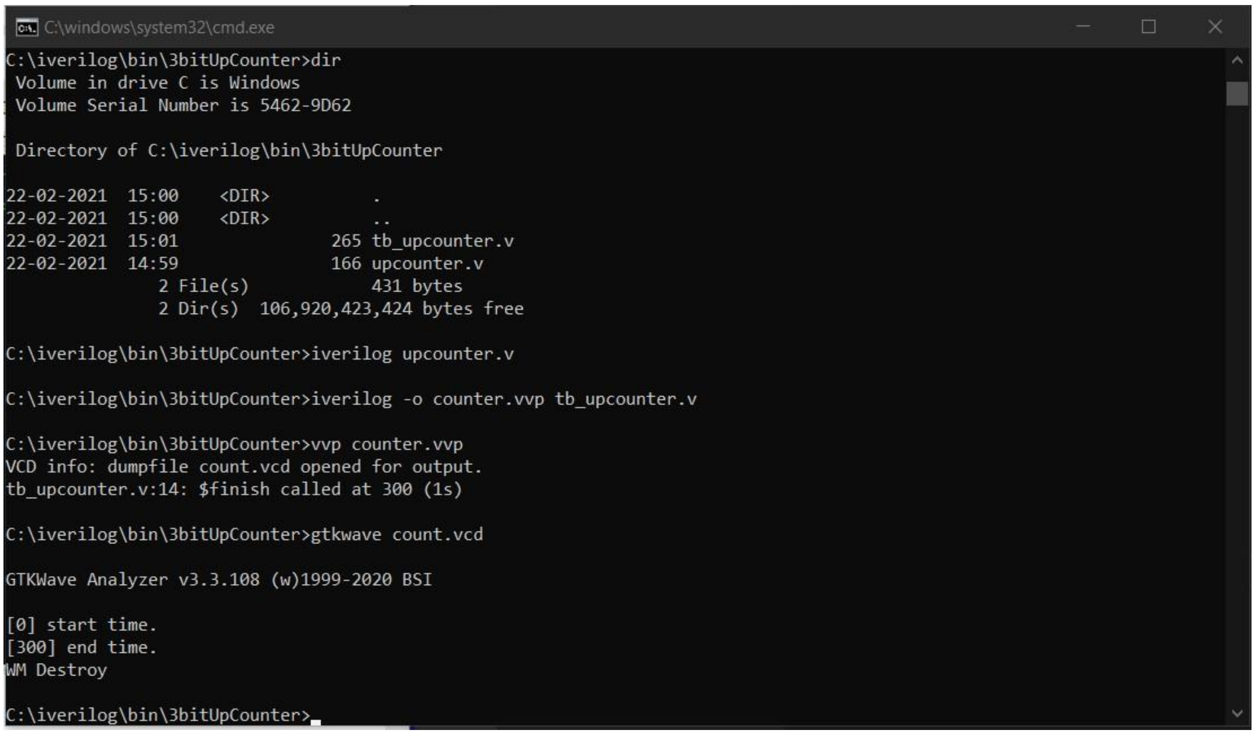
Write HDL implementation for a 3-bit up-counter using behavioral model. Simulate the same using behavioral model and depict the timing diagram for valid inputs.

Main Module (upcounter.v)

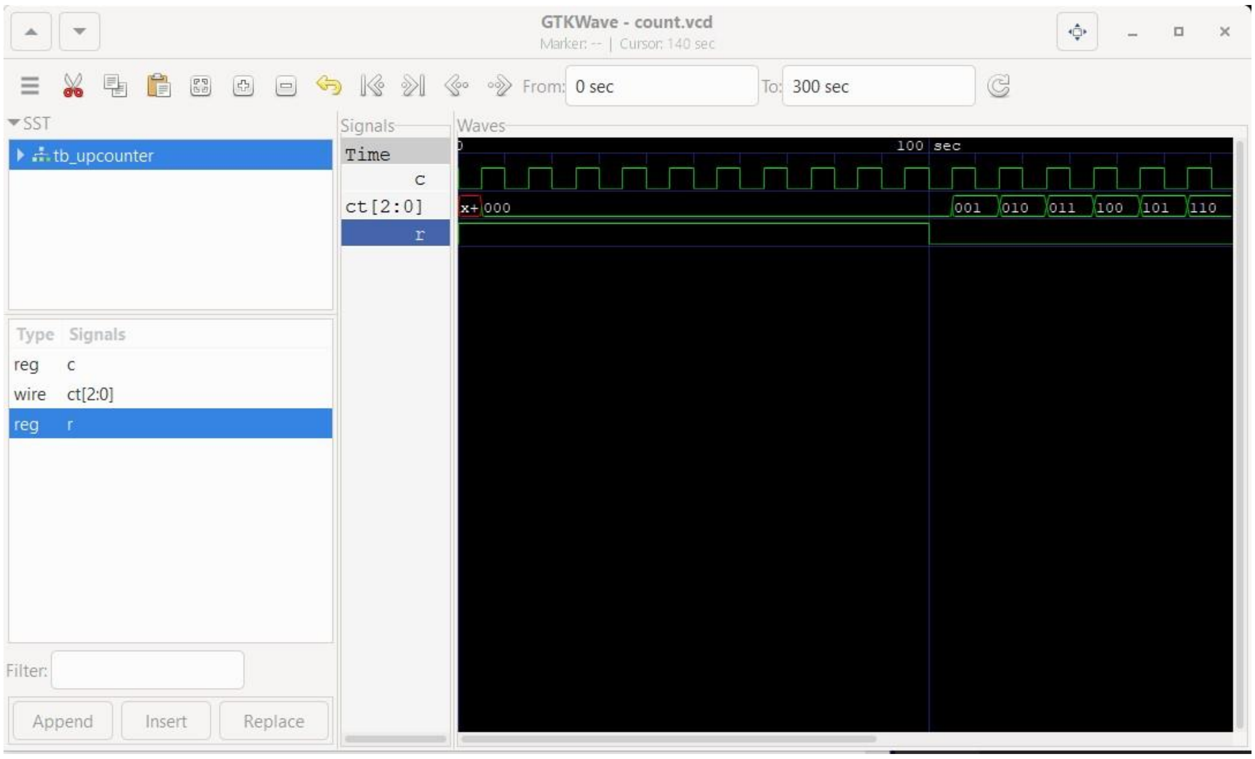
```
module upcounter( count,rst,clk);
input rst, clk;
output reg [2:0] count;
always @(posedge (clk))
if (rst)
count<= 3'b000;
else
count<= count + 1;
endmodule
```

TEST MODULE (tb_upcounter.v)

```
`include "upcounter.v"
module tb_upcounter;
reg r,c;
wire  [2:0] ct;


upcounter countbeh (ct,r,c);
initial
begin
$dumpfile("count.vcd");
$dumpvars(0,tb_upcounter);
r =1;
c=0;
#100 r=0;
#200 $finish;
end
always #5 c=~c;
endmodule
```



```
C:\iverilog\bin\3bitUpCounter>dir
 Volume in drive C is Windows
 Volume Serial Number is 5462-9D62

 Directory of C:\iverilog\bin\3bitUpCounter

22-02-2021  15:00    <DIR>          .
22-02-2021  15:00    <DIR>          ..
22-02-2021  15:01               265 tb_upcounter.v
22-02-2021  14:59               166 upcounter.v
               2 File(s)            431 bytes
               2 Dir(s)  106,920,423,424 bytes free

C:\iverilog\bin\3bitUpCounter>iverilog upcounter.v

C:\iverilog\bin\3bitUpCounter>iverilog -o counter.vvp tb_upcounter.v

C:\iverilog\bin\3bitUpCounter>vvp counter.vvp
VCD info: dumpfile count.vcd opened for output.
tb_upcounter.v:14: $finish called at 300 (1s)

C:\iverilog\bin\3bitUpCounter>gtkwave count.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[300] end time.
WM Destroy

C:\iverilog\bin\3bitUpCounter>
```



Type	Signals
reg	c
wire	ct[2:0]
reg	r

DATA FLOW MODELING

Experiment 11

Write HDL implementation for AND/OR/NOT gates using data flow model. Simulate the same using data flow model and depict the timing diagram for valid inputs.

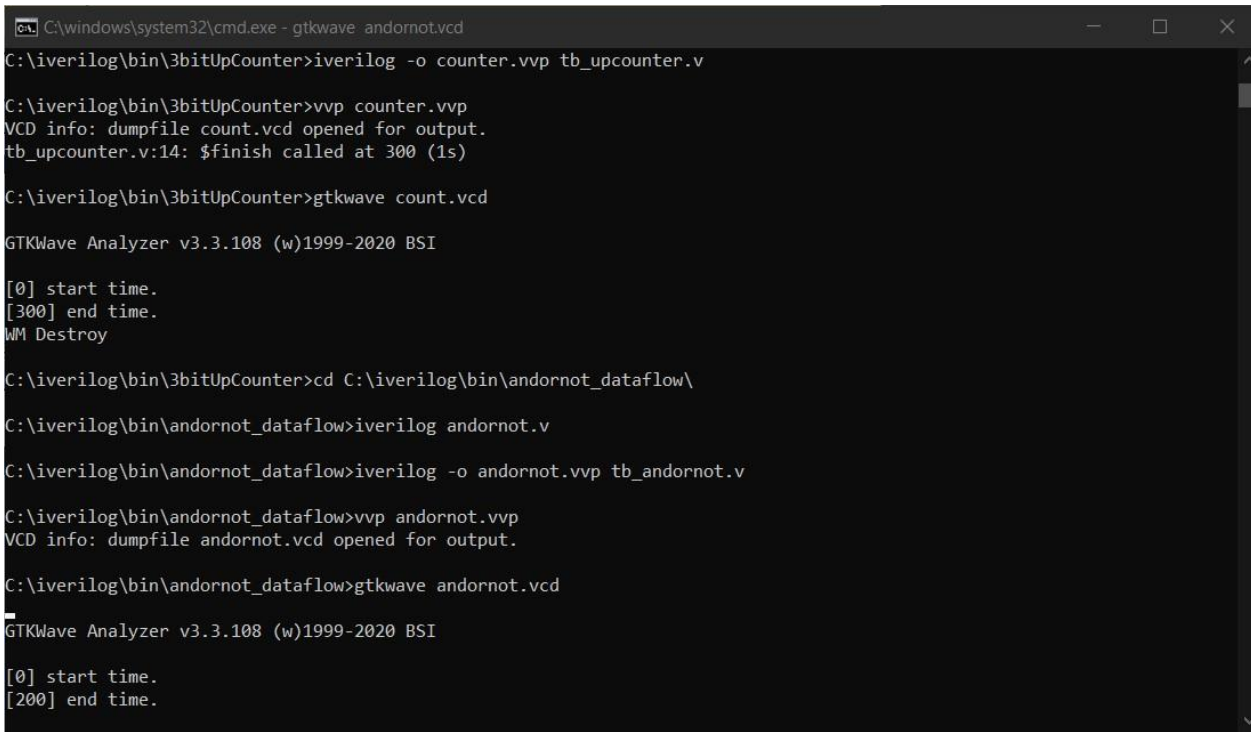
MAIN MODULE (andornot.v)

```
module andornot(input a,b,output [2:0]y);

assign y[2]=a&b;

assign y[1]=a|b;

assign y[0]=~a;

endmodule
```

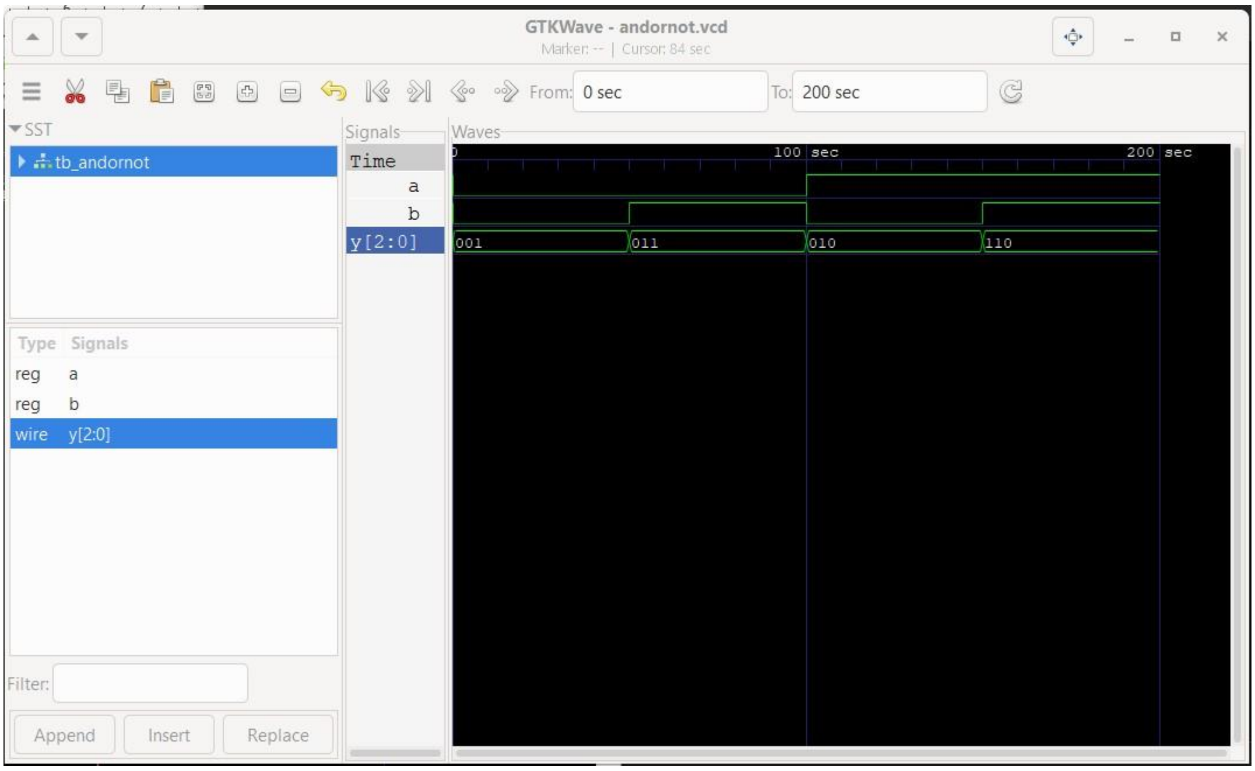
TESTBENCH MODULE (tb_andornot.v)

```
`include "andornot.v"

module tb_andornot;

wire [2:0]y;

reg a,b;

andornot dut(.y(y),.a(a),.b(b));

initial

begin

$dumpfile("andornot.vcd");

$dumpvars(0,tb_andornot);

a=1'b0;

b=1'b0;

#50;

a=1'b0;

b=1'b1;

#50;

a=1'b1;

b=1'b0;
```

```
#50;

a=1'b1;

b=1'b1;

#50;

end

endmodule
```



```
C:\iverilog\bin\3bitUpCounter>iverilog -o counter.vvp tb_upcounter.v

C:\iverilog\bin\3bitUpCounter>vvp counter.vvp
VCD info: dumpfile count.vcd opened for output.
tb_upcounter.v:14: $finish called at 300 (1s)

C:\iverilog\bin\3bitUpCounter>gtkwave count.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[300] end time.
WM Destroy

C:\iverilog\bin\3bitUpCounter>cd C:\iverilog\bin\andornot_dataflow\

C:\iverilog\bin\andornot_dataflow>iverilog andornot.v

C:\iverilog\bin\andornot_dataflow>iverilog -o andornot.vvp tb_andornot.v

C:\iverilog\bin\andornot_dataflow>vvp andornot.vvp
VCD info: dumpfile andornot.vcd opened for output.

C:\iverilog\bin\andornot_dataflow>gtkwave andornot.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

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



Experiment 12

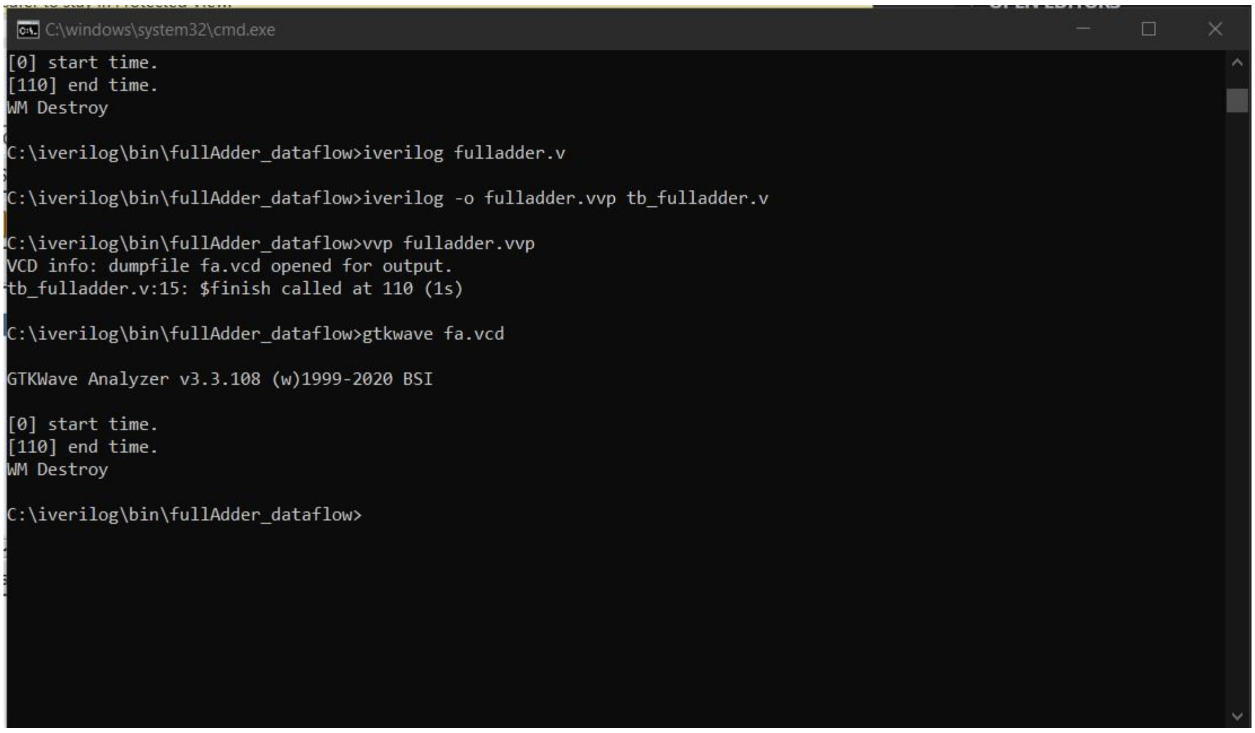
Write HDL implementation for a 3-bit full adder using data flow model. Simulate the same using data flow model and depict the timing diagram for valid inputs.

MAIN MODULE (fulladder.v)

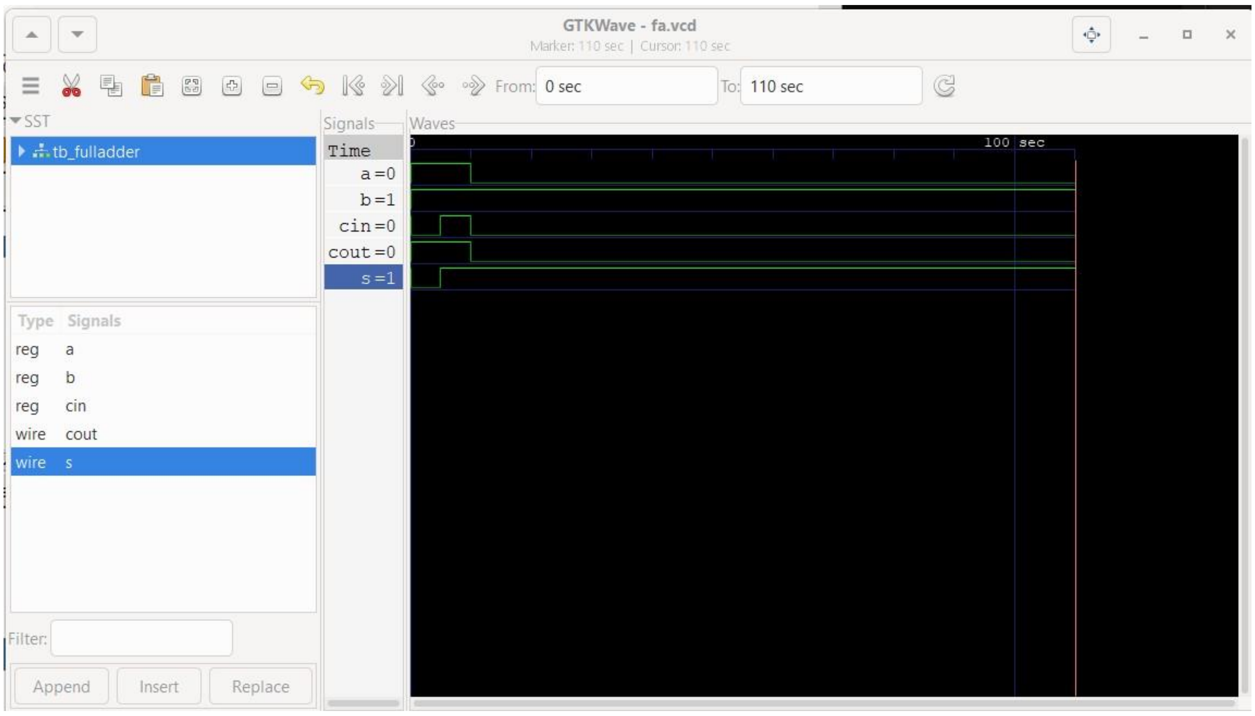
```
module fulladder(a,b,cin,s,cout);
input a,b,cin;
output s,cout;
assign s=a^b^cin;
assign cout=(a&b)|(b&cin)|(cin&a);
endmodule
```

TEST MODULE (tb_fulladder.v)

```
`include "fulladder.v"

module tb_fulladder;

reg a,b,cin;

wire s,cout;

fulladder f1(a,b,cin,s,cout);

initial

begin

$dumpfile("fa.vcd");

$dumpvars(0,tb_fulladder);

                a=1;b=1;cin=0;

                #5

                a=1;b=1;cin=1;

                #5

                a=0;b=1;cin=0;

                #100 $finish;

                end

endmodule
```



```
[0] start time.
[110] end time.
WM Destroy

C:\iverilog\bin\fullAdder_dataflow>iverilog fulladder.v

C:\iverilog\bin\fullAdder_dataflow>iverilog -o fulladder.vvp tb_fulladder.v

C:\iverilog\bin\fullAdder_dataflow>vvp fulladder.vvp
VCD info: dumpfile fa.vcd opened for output.
tb_fulladder.v:15: $finish called at 110 (1s)

C:\iverilog\bin\fullAdder_dataflow>gtkwave fa.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[110] end time.
WM Destroy

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



```
Time
a=0
b=1
cin=0
cout=0
s=1
```

Type	Signals
reg	a
reg	b
reg	cin
wire	cout
wire	s

Signature of the staff in-charge