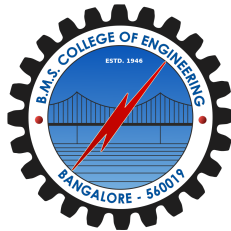


B.M.S College of Engineering
(Autonomous Institution affiliated to VTU, Belagavi)
Bengaluru – 19

Department of Computer Science and Engineering



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

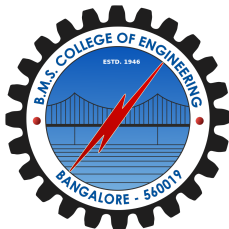
LOGIC DESIGN - 19CS3PCLOD

(Autonomous Scheme 2019)

Submitted by

Name: Rakshith R
USN:1BM20CS123

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. **Rakshith R** has satisfactorily completed the course of Experiments in **LOGIC DESIGN** course prescribed by the Department during the year **2021-2022**

Name of the Candidate: **Rakshith R**

USN No.: 1BM20CS123 Semester: **3**

Marks	
Max. Marks	Obtained
10	

Marks in Words	

Signature of the staff in-charge
Date: 18/01/2022

Head of the Department

Verilog Program List

19CS3PCLOD

Serial No.	Title
	CYCLE I Structural Modeling
•	<p>Write HDL implementation for the following Logic</p> <ul style="list-style-type: none"> • AND/OR/NOT <p>Simulate the same using structural model and depict the timing diagram for valid inputs.</p>
•	<p>Write HDL implementation for the following Logic</p> <ul style="list-style-type: none"> • NAND/NOR <p>Simulate the same using structural model and depict the timing diagram for valid inputs.</p>
•	<p>Write HDL implementation for the following Boolean expression:</p> $A'C' + A'C + AB'$ <p>Simulate the same using structural model and depict the timing diagram for valid inputs.</p>
•	<p>Write HDL implementation for a 8:1 Multiplexer. Simulate the same using structural model and depict the timing diagram for valid inputs.</p>
•	<p>Write HDL implementation for a 2-to-4 decoder. Simulate the same using structural model and depict the timing diagram for valid inputs.</p>
• 8	<p>Write HDL implementation for a 4-to-2 encoder. Simulate the same using structural model and depict the timing diagram for valid inputs.</p>

	<p style="text-align: center;">CYCLE II Behavior Modeling</p>
•	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 down-counter using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
	<p style="text-align: center;">CYCLE III Dataflow Modeling</p>
•	Write HDL implementation for NAND/NOR/EXOR 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

```
module s_andornot(a,b,y);  
    input a,b;  
    output [2:0]y;  
    and ag(y[2],a,b);  
    or og(y[1],a,b);  
    not ng(y[0],a);  
endmodule
```

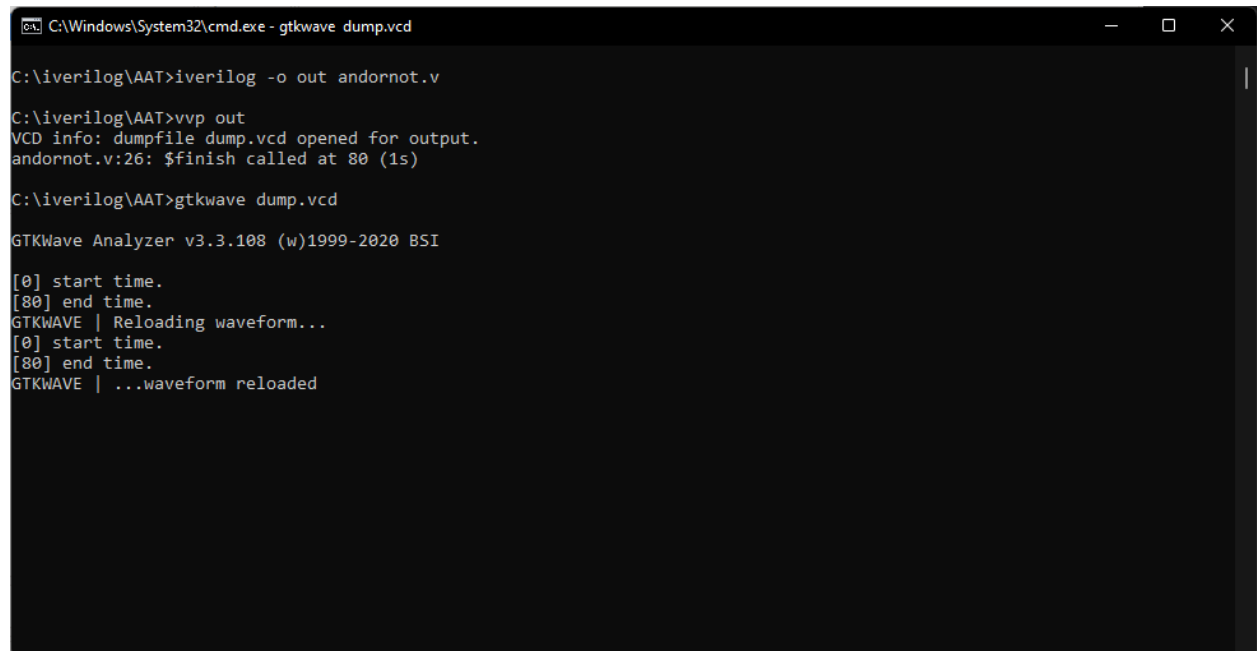
TESTBENCH MODULE

```
module tb_andornot;  
    reg a,b;  
    wire [2:0]y;  
    s_andornot ob(a,b,y);  
    initial  
    begin  
        $dumpfile("dump.vcd");  
        $dumpvars(0,tb_andornot);  
        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
```

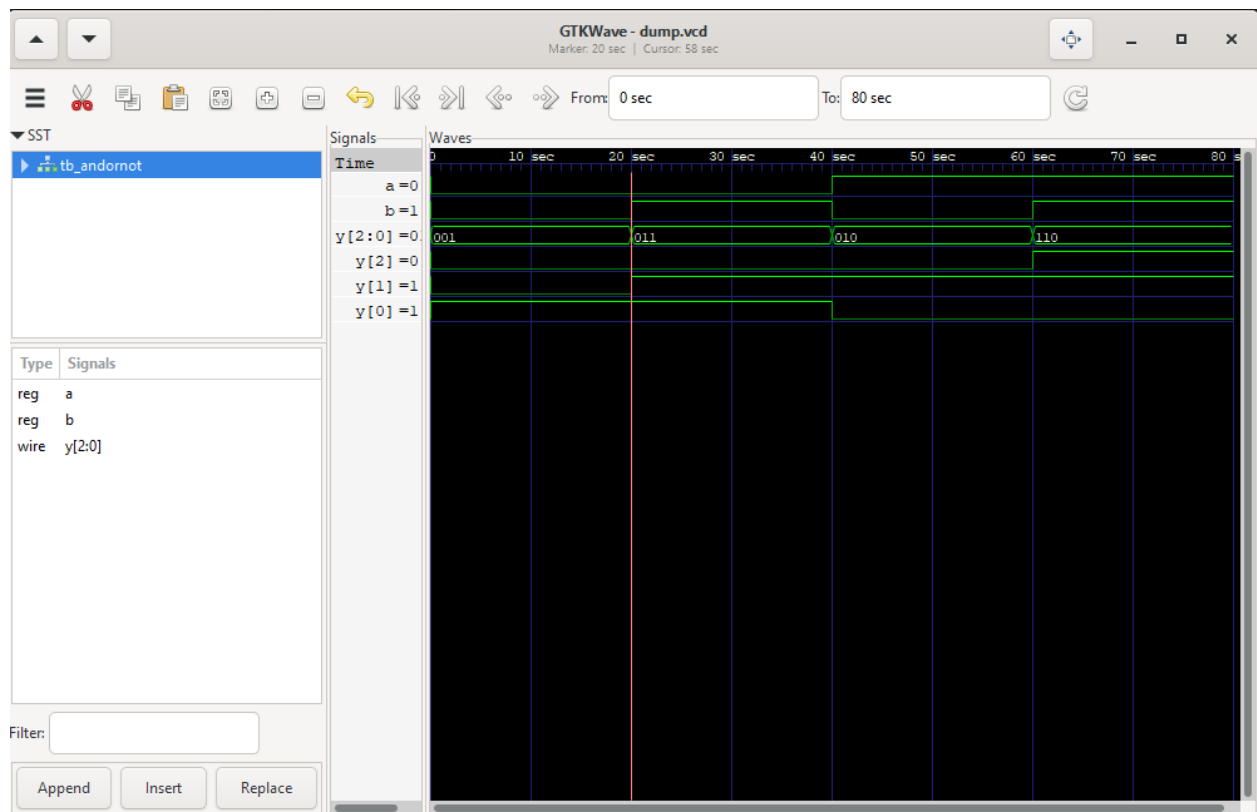
OUTPUT:



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd

C:\iverilog\AAT>iverilog -o out andornot.v
C:\iverilog\AAT>vvp out
VCD info: dumpfile dump.vcd opened for output.
andornot.v:26: $finish called at 80 (1s)
C:\iverilog\AAT>gtkwave dump.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI
[0] start time.
[80] end time.
GTKWAVE | Reloading waveform...
[0] start time.
[80] end time.
GTKWAVE | ...waveform reloaded
```



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

```
module no( t0 , a , b , t1);
    input a , b ;
    output t0 , t1 ;
    and a1(x1 , a , b );
    not n1(t0 , x1 );
    or o1 (x2 , a , b);
    not n2(t1 , x2 );
endmodule
```

TESTBENCH MODULE

```
module testbench;
    wire t0 , t1;
    reg a , b;
    no g(t0 , a , b , t1);
    initial
    begin
        $dumpfile("nand_nor.vcd");
        $dumpvars( 0 , testbench );
        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
```



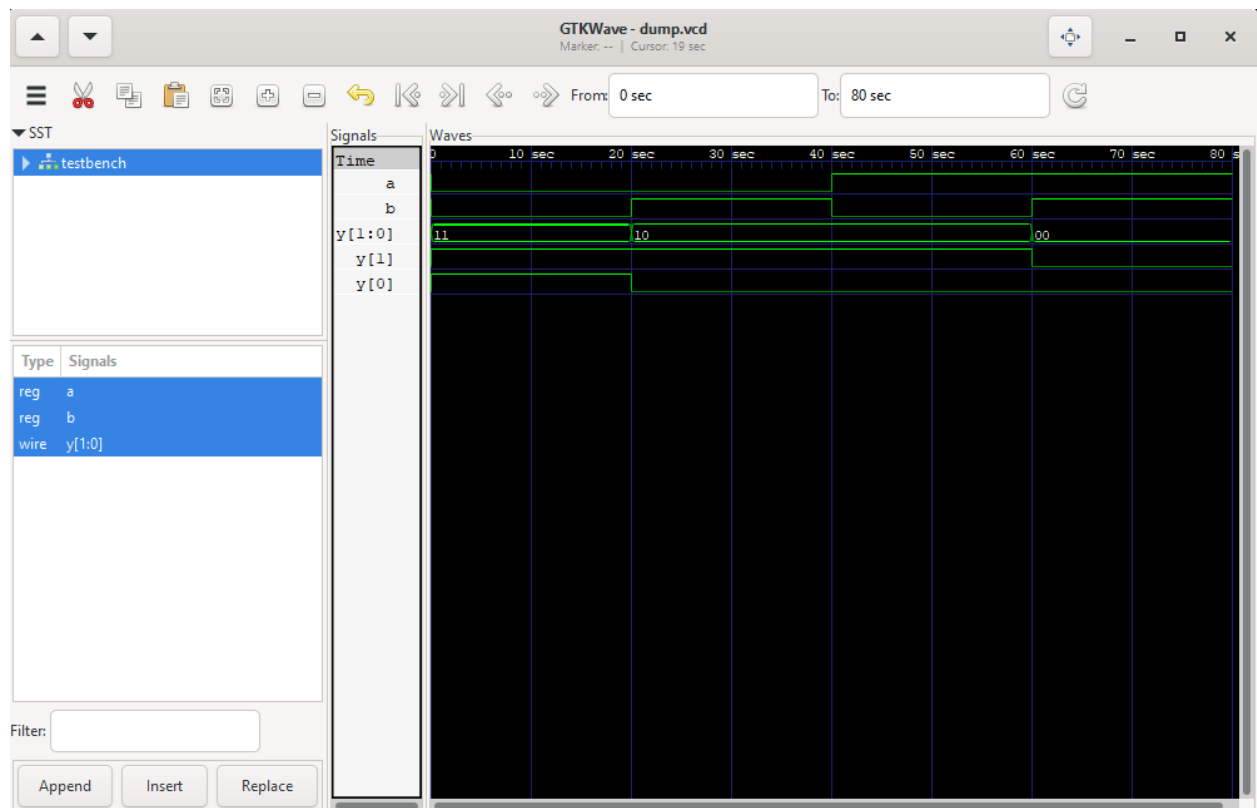
```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd

C:\iverilog\AAT\nandnor>iverilog -o out nandnor.v
C:\iverilog\AAT\nandnor>vvp out
VCD info: dumpfile dump.vcd opened for output.
nandnor.v:26: $finish called at 80 (1s)

C:\iverilog\AAT\nandnor>gtkwave dump.vcd

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

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



Experiment 3

Write HDL implementation for the following Boolean expression:

$$A'C' + A'C + AB'$$

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

MAIN MODULE

```
module addor(a,b,c,y);
    input a,b,c;
    output y;
    wire and_op1, and_op2, and_op3, and_op4, and_op5;
    and g1(and_op1,~a,~c);
    and g2(and_op2,~a,c);
    and g3(and_op3,a,~b);

    or g4(and_op4, and_op1, and_op2);
    or g5(and_op5, and_op3, and_op4);
    or g6(y, and_op5, and_op4);

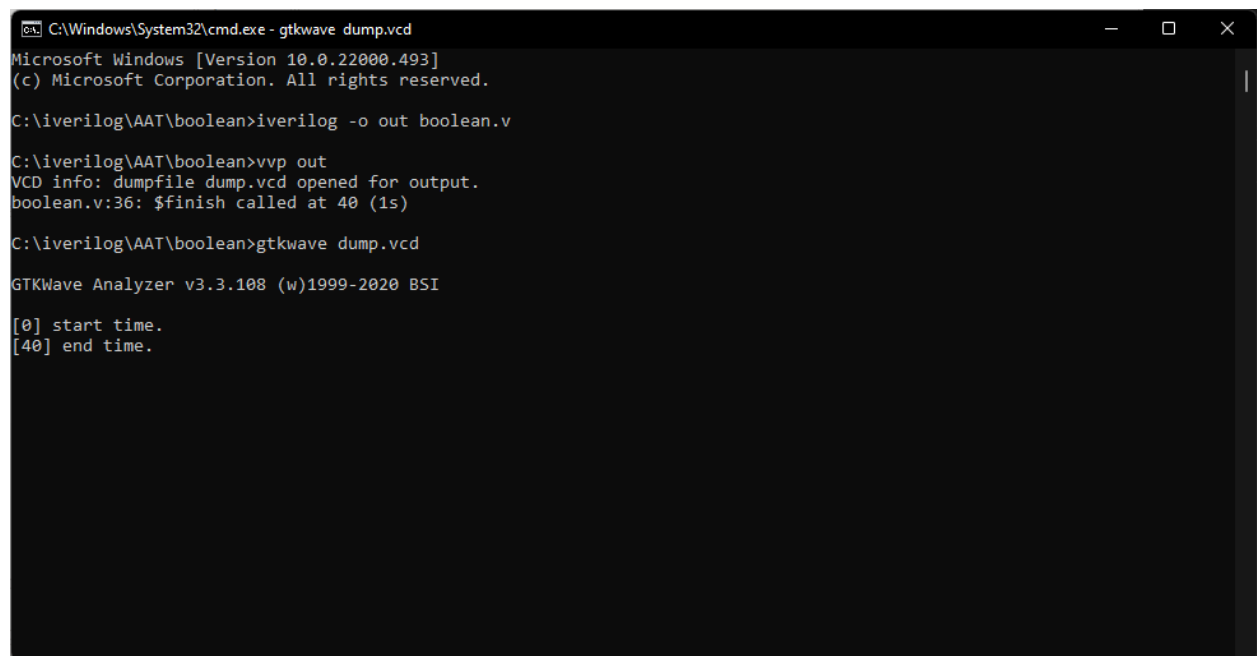
endmodule
```

TESTBENCH MODULE

```
module testbench;
    reg a,b,c,d;
    wire y;
    addor ao(a,b,c,y);
    initial begin
        $dumpfile("and_or.vcd");
        $dumpvars(0,testbench);
        a=0;b=0;c=0;
    end
endmodule
```

```
    #5
    a=0;b=0;c=1;
    #5
    a=0;b=1;c=0;
    #5
    a=0;b=1;c=1;
    #5
    a=1;b=0;c=0;
    #5
    a=1;b=0;c=1;
    #5
    a=1;b=1;c=0;
    #5
    a=1;b=1;c=1;
    $finish;
end
```

```
endmodule
```



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

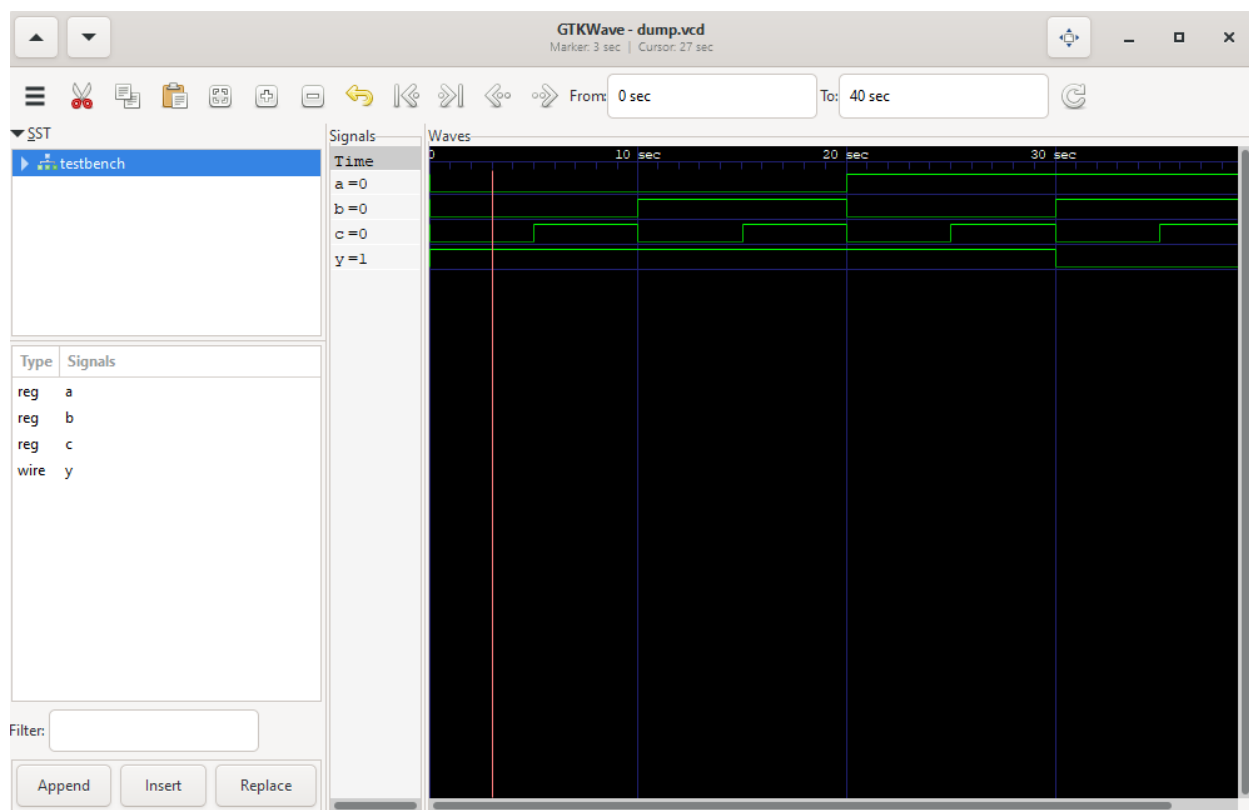
C:\iverilog\AAT\boolean>iverilog -o out boolean.v

C:\iverilog\AAT\boolean>vvp out
VCD info: dumpfile dump.vcd opened for output.
boolean.v:36: $finish called at 40 (1s)

C:\iverilog\AAT\boolean>gtkwave dump.vcd

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

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



Experiment 4

Write HDL implementation for a 8:1 Multiplexer. Simulate the same using structural model and depict the timing diagram for valid inputs

MAIN MODULE

```
module m81(out, D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2);
    output out;
    input D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2;
    wire s0bar, s1bar, T1, T2, T3, T4, T5, T6, T7, T8;

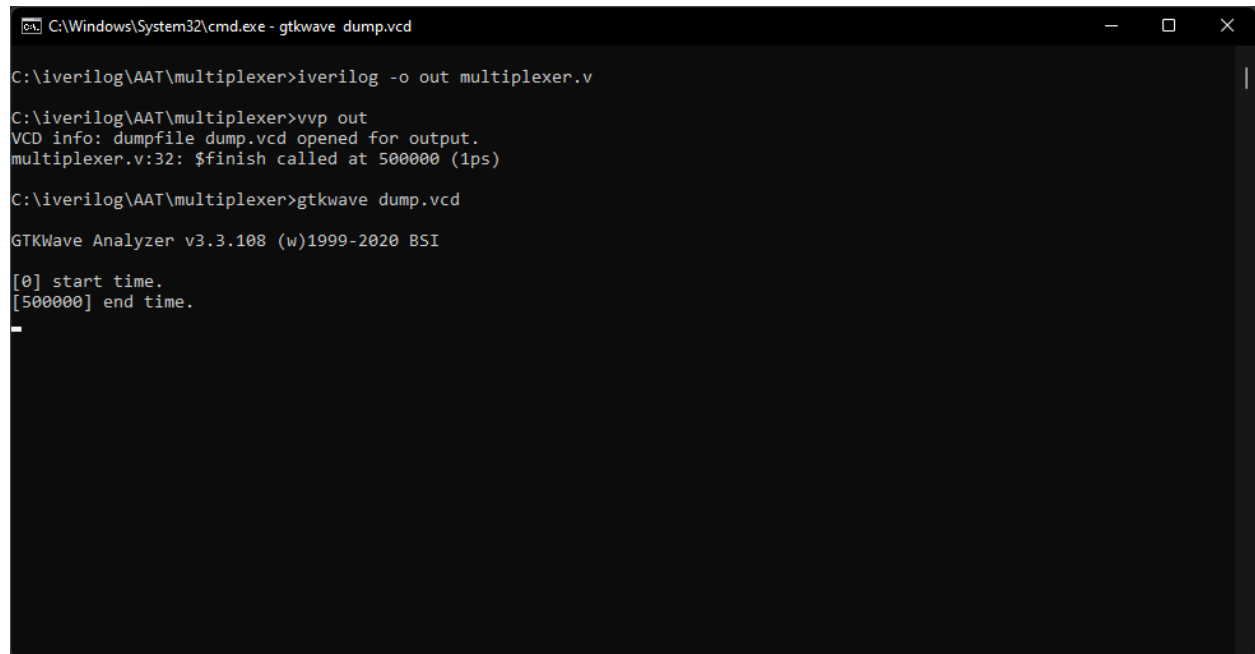
    not u1(s1bar, S1);
    not u2(s0bar, S0);
    not u3(s2bar, S2);

    and u4(T1, D0, s0bar, s1bar, s2bar);
    and u5(T2, D1, S0, s1bar, s2bar);
    and u6(T3, D2, s0bar, S1, s2bar);
    and u7(T4, D3, S0, S1, s2bar);
    and u8(T5, D4, s0bar, s1bar, S2);
    and u9(T6, D5, S0, s1bar, S2);
    and u10(T7, D6, s0bar, S1, S2);
    and u11(T8, D7, S0, S1, S2);
    or u12(out, T1, T2, T3, T4, T5, T6, T7, T8);
endmodule
```

TESTBENCH MODULE

```
`timescale 1ns/1ps
module top;
    wire out;
    reg D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2;
    m81 name(.D0(D0), .D1(D1), .D2(D2), .D3(D3), .D4(D4), .D5(D5), .D6(D6), .D7(D7),
    .S0(S0), .S1(S1), .S2(S2), .out(out));
    initial
    begin
        $dumpfile("dump.vcd");
        $dumpvars(0 , top);
        D0=1'b0; D1=1'b0; D2=1'b0; D3=1'b0; D4=1'b0; D5=1'b0; D6=1'b0; D7=1'b0; S0=1'b0;
        S1=1'b0; S2=1'b0;
```

```
#500 $finish;
end
always #1 D0=~D0;
always #2 D1=~D1;
always #3 D2=~D2;
always #4 D3=~D3;
always #5 D4=~D4;
always #6 D5=~D5;
always #7 D6=~D6;
always #8 D7=~D7;
always #9 S0=~S0;
always #10 S1=~S1;
always #11 S2=~S2;
endmodule;
```



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd

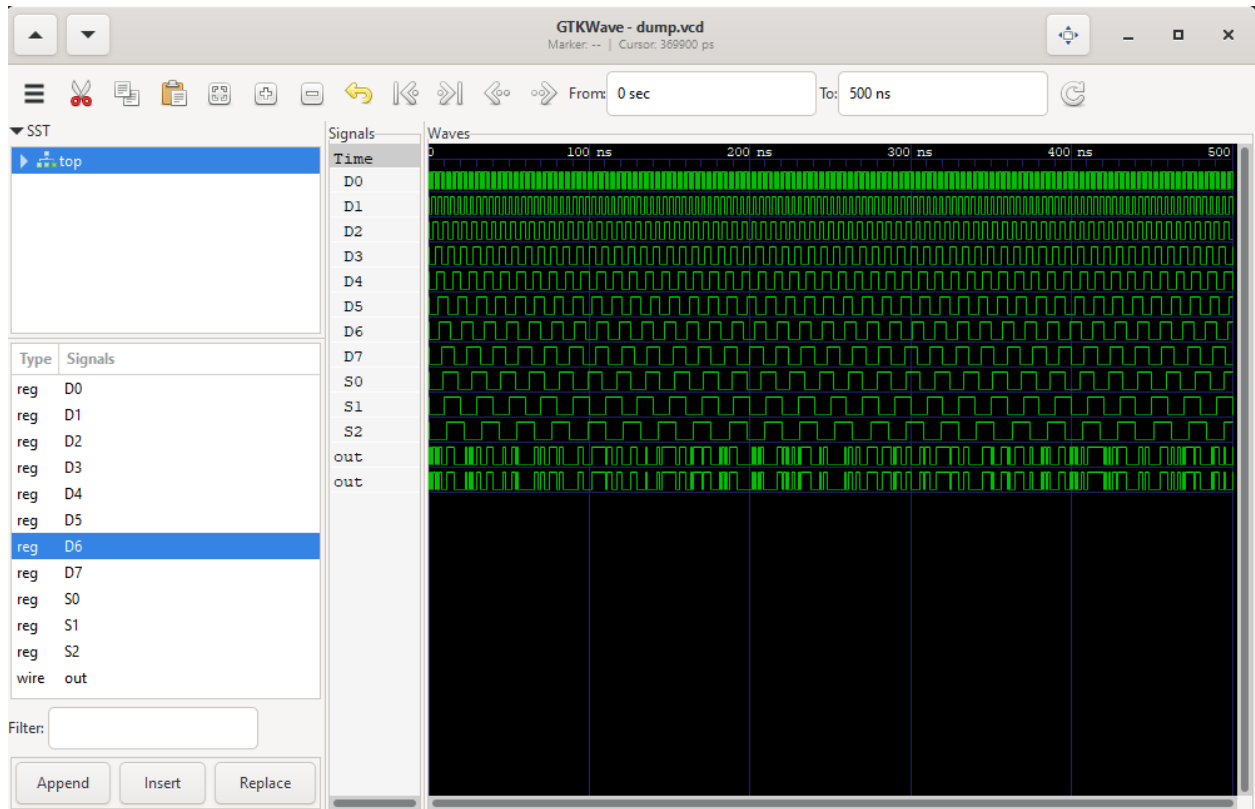
C:\iverilog\AAT\multiplexer>iverilog -o out multiplexer.v

C:\iverilog\AAT\multiplexer>vvp out
VCD info: dumpfile dump.vcd opened for output.
multiplexer.v:32: $finish called at 500000 (1ps)

C:\iverilog\AAT\multiplexer>gtkwave dump.vcd

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

[0] start time.
[500000] 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

```
module bcddecoder2to4(b0 , b1 , d0 , d1 , d2 , d3 );  
    input b0 , b1;  
    output d0 , d1 , d2 , d3 ;  
    wire t0 , t1;  
    not n1(t0 , b0);  
    not n2( t1 , b1);  
    and a0(d0 , t0 , t1);  
    and a1(d1 , t0 , b1);  
    and a2(d2 , b0 , t1);  
    and a3(d3 , b0 , b1);  
endmodule
```

TESTBENCH MODULE

```
module test;  
    reg b0 , b1;  
    wire d0 , d1 , d2 , d3 ;  
    bcddecoder2to4 bcdg(b0 , b1 , d0 , d1 , d2 , d3);  
    initial  
    begin  
        $dumpfile("dump.vcd");  
        $dumpvars( 0 ,test);  
        b0 = 0; b1 = 0;  
        #40  
        b0 = 0; b1 = 1;
```


#40

b0 = 1; b1 = 0;

#40

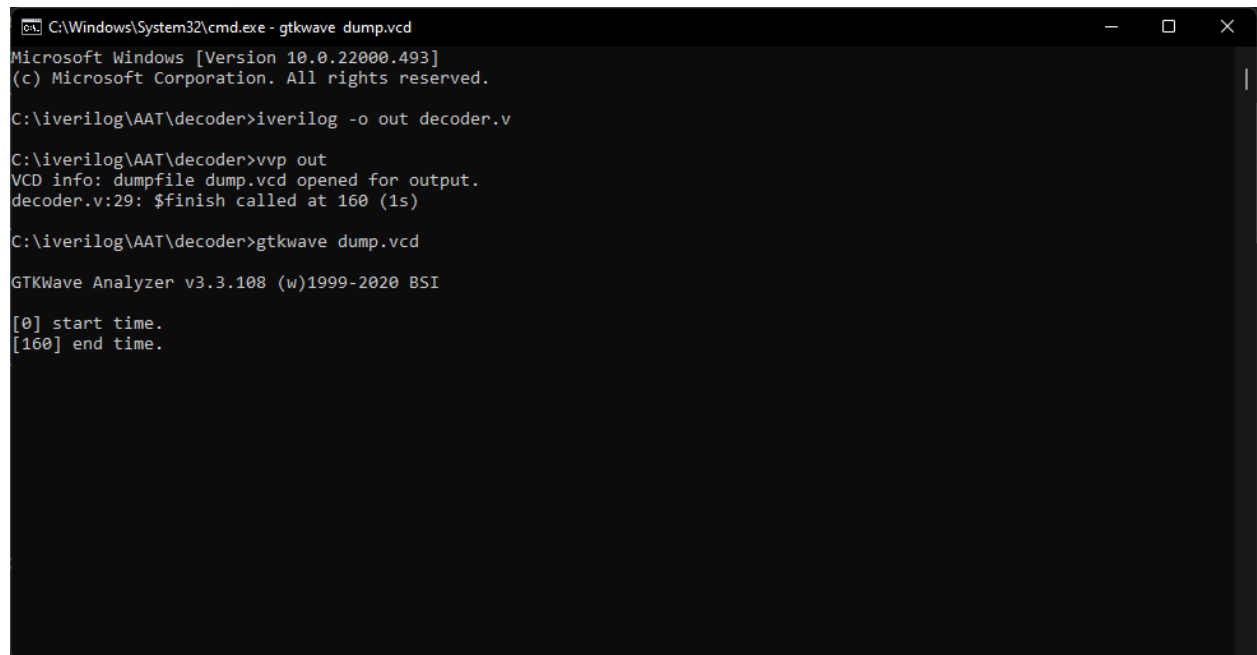
b0 = 1; b1 = 1;

#40

\$finish;

end

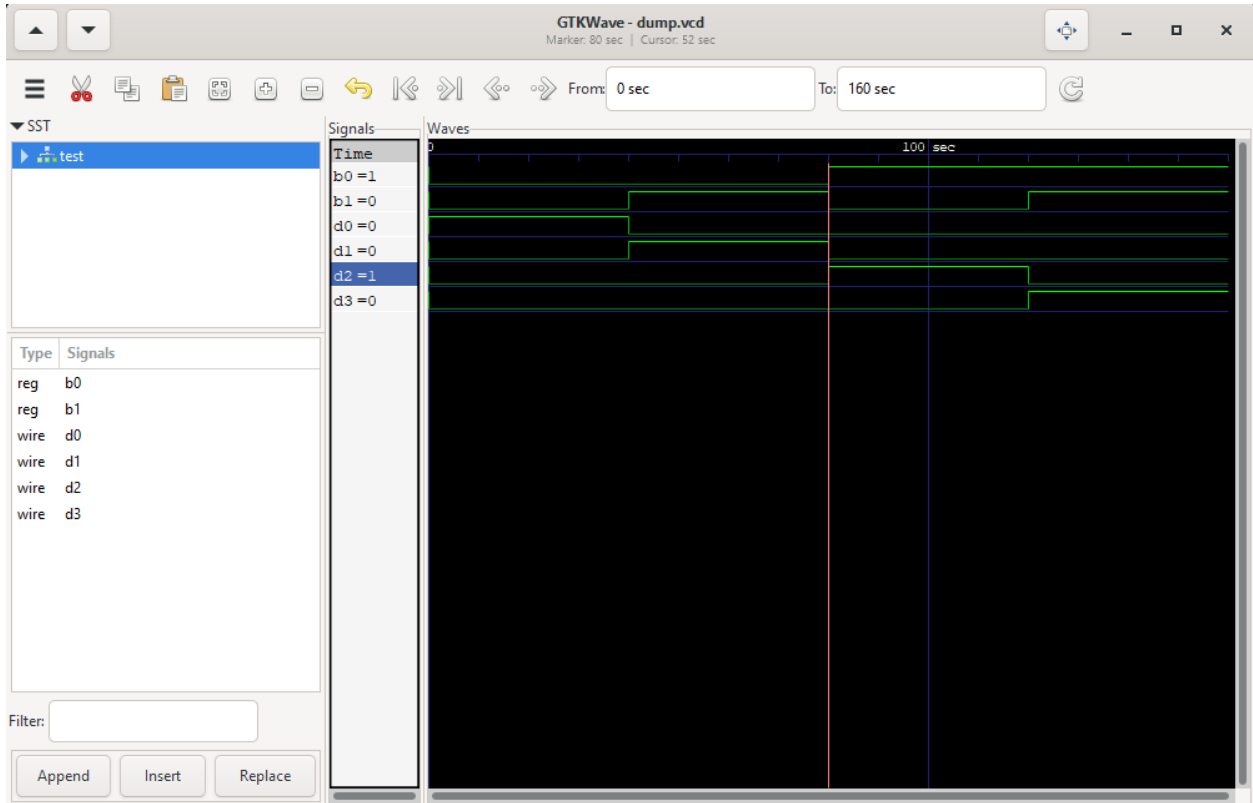
endmodule



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\iverilog\AAT\decoder>iverilog -o out decoder.v
C:\iverilog\AAT\decoder>vvp out
VCD info: dumpfile dump.vcd opened for output.
decoder.v:29: $finish called at 160 (1s)
C:\iverilog\AAT\decoder>gtkwave dump.vcd
GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

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



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.

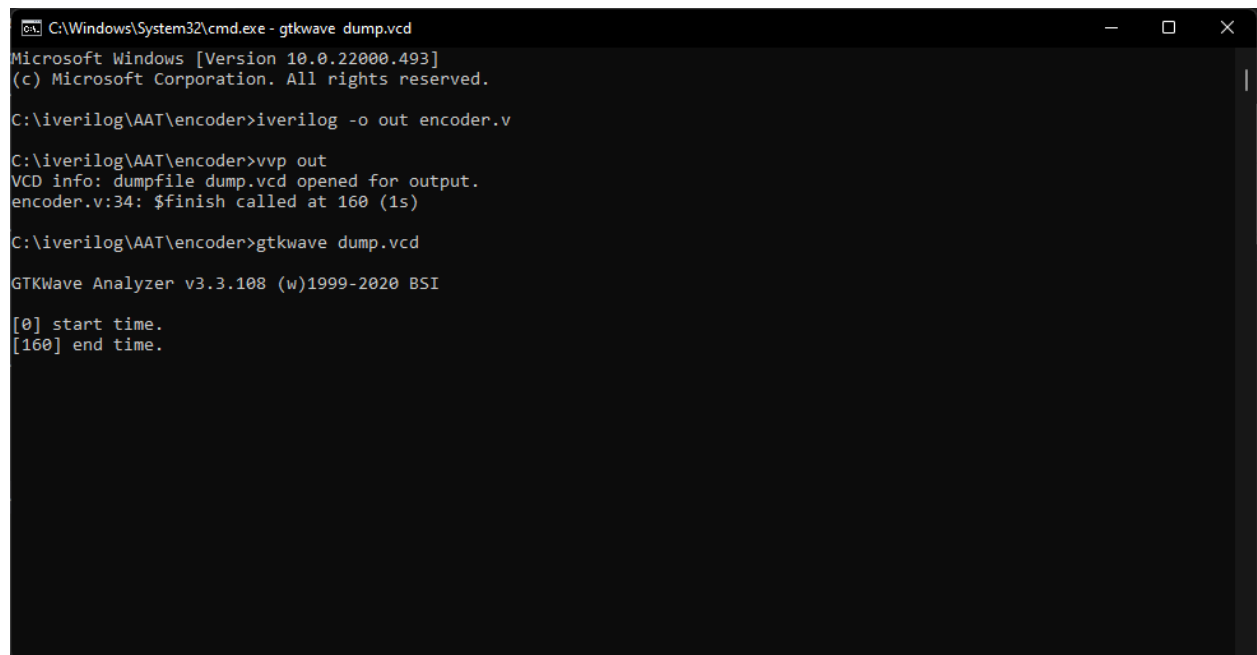
MAIN MODULE

```
module encoder4to2( b0 , b1 , d0 , d1 , d2 , d3);  
    output b0 , b1;  
    input d0 , d1 , d2 , d3 ;  
    wire t0 , t1 , t2 , t3 , t4 , t5 , t6 , t7;  
    not n0( t0 , d0);  
    not n1( t1 , d1);  
    not n2( t2 , d2);  
    not n3( t3 , d3);  
    and a0( t4 , t0 , t1 , d2 , t3);  
    and a1( t5 , t0 , t1 , t2 , d3);  
    and a2( t6 , t0 , d1 , t2 , t3);  
    and a3( t7 , t0 , t1 , t2 , d3);  
    or o0(b0 , t4, t5);  
    or o1(b1 , t6 , t7);  
endmodule
```

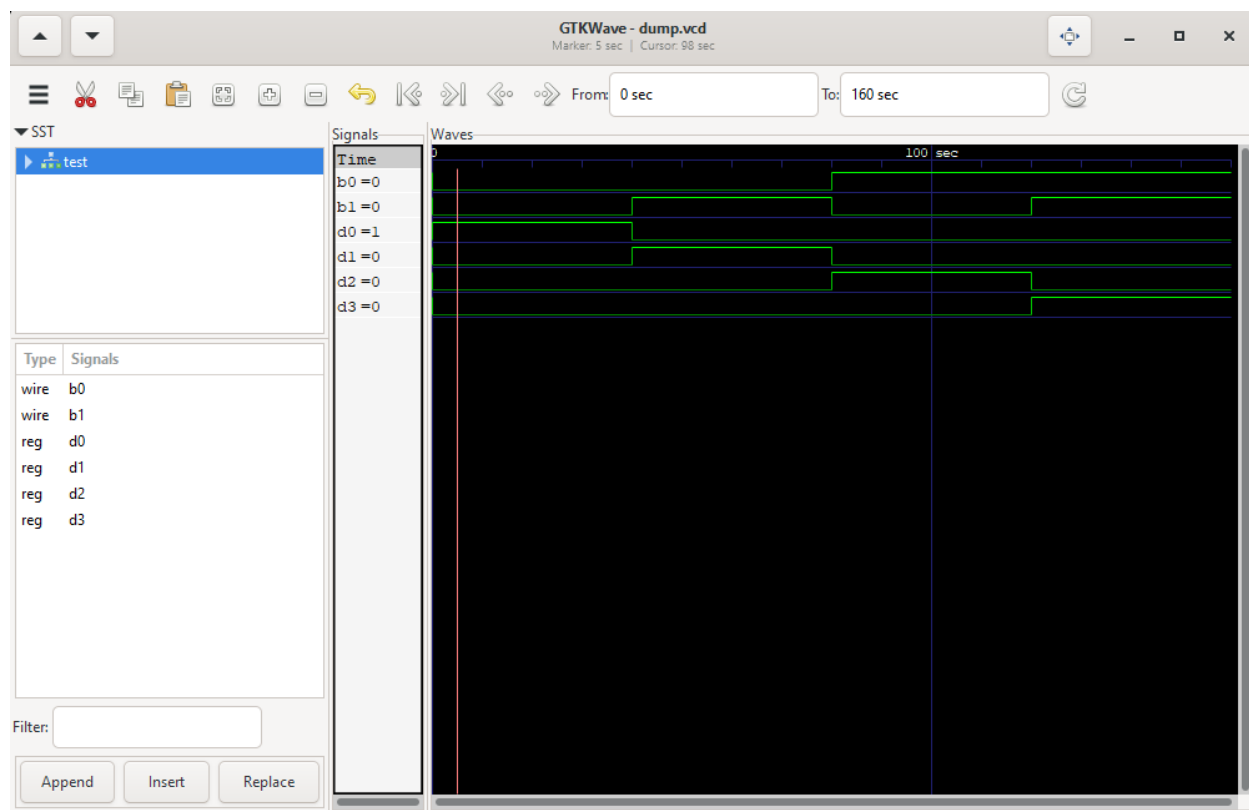
TESTBENCH MODULE

```
module test ;  
    wire b0 , b1 ;  
    reg d0 , d1 , d2 , d3;  
    encoder4to2 encg( b0 , b1 , d0 , d1 , d2 , d3 );  
    initial  
    begin
```

```
$dumpfile("dump.vcd");  
$dumpvars(0 , test);  
  
d0 = 1 ; d1 = 0; d2 = 0 ; d3 = 0;  
#40  
  
d0 = 0 ; d1 = 1; d2 = 0 ; d3 = 0;  
#40  
  
d0 = 0 ; d1 = 0; d2 = 1 ; d3 = 0;  
#40  
  
d0 = 0 ; d1 = 0; d2 = 0 ; d3 = 1;  
#40  
  
$finish;  
  
end  
  
endmodule
```



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd  
Microsoft Windows [Version 10.0.22000.493]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\iverilog\AAT\encoder>iverilog -o out encoder.v  
  
C:\iverilog\AAT\encoder>vvp out  
VCD info: dumpfile dump.vcd opened for output.  
encoder.v:34: $finish called at 160 (1s)  
  
C:\iverilog\AAT\encoder>gtkwave dump.vcd  
  
GTKWave Analyzer v3.3.108 (w)1999-2020 BSI  
  
[0] start time.  
[160] end time.
```



CYCLE II Behavior Modeling

Experiment 7

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.

MAIN MODULE

```
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
```

TESTBENCH MODULE

```
module test_srflipf;
```

```
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("dump.vcd");

    $dumpvars(0 , test_srflipf);

    A=2'b00;#10;

    A=2'b01;#10;

    A=2'b10;#10;

    A=2'b11;

    #20

    $finish;

end

endmodule
```

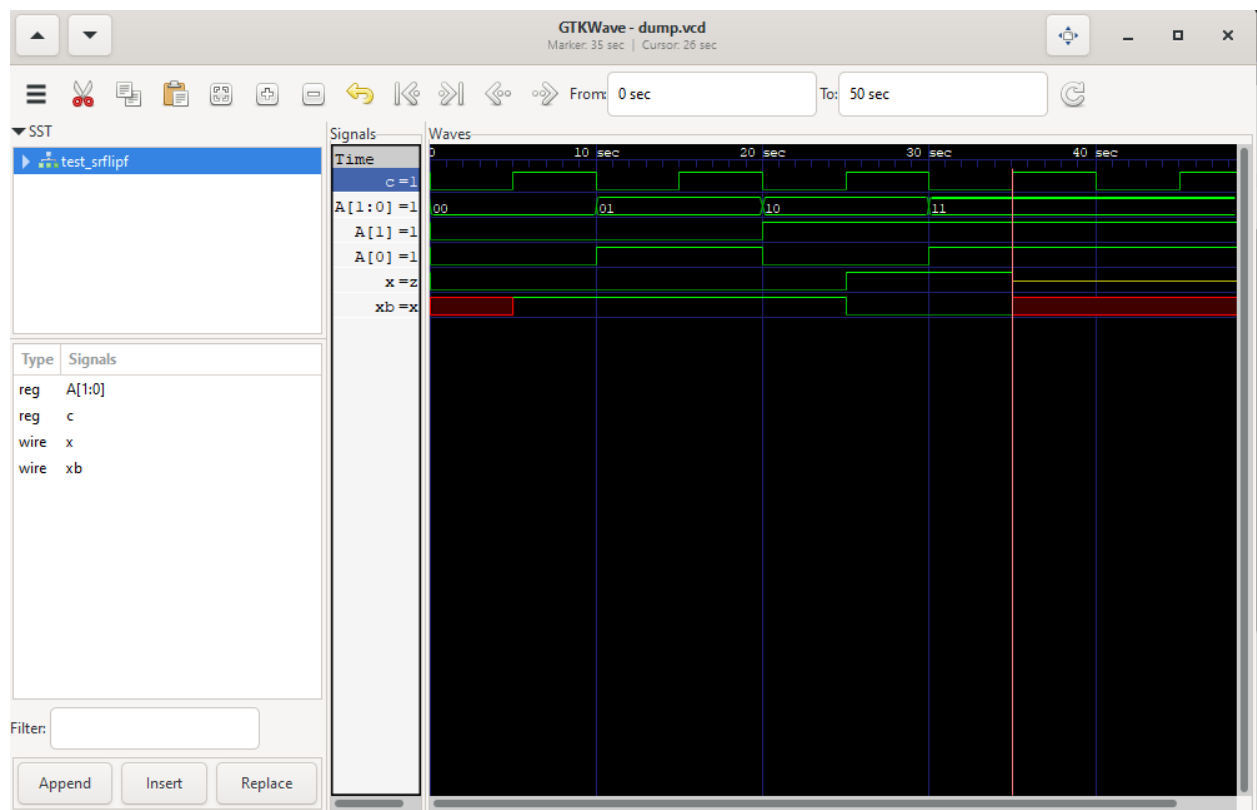
```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\liverilog\AAT\rs-ff>liverilog -o out rs-ff.v
C:\liverilog\AAT\rs-ff>vvp out
VCD info: dumpfile dump.vcd opened for output.
rs-ff.v:35: $finish called at 50 (1s)

C:\liverilog\AAT\rs-ff>gtkwave dump.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 Behavior model and depict the timing diagram for valid inputs.

MAIN MODULE

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

TESTBENCH MODULE

```
module test_jkflipf;  
    reg[1:0]A;  
    reg c;
```

```
wire x , xb;

JK_FF jkff(A , c , x, xb);

initial c=1'b0;

always #5 c=~c;

initial

begin

    $dumpfile("dump.vcd");

    $dumpvars(0 , test_jkflipf);

    A=2'b00;#10;

    A=2'b01;#10;

    A=2'b10;#10;

    A=2'b11;

    #20

    $finish;

end

endmodule
```

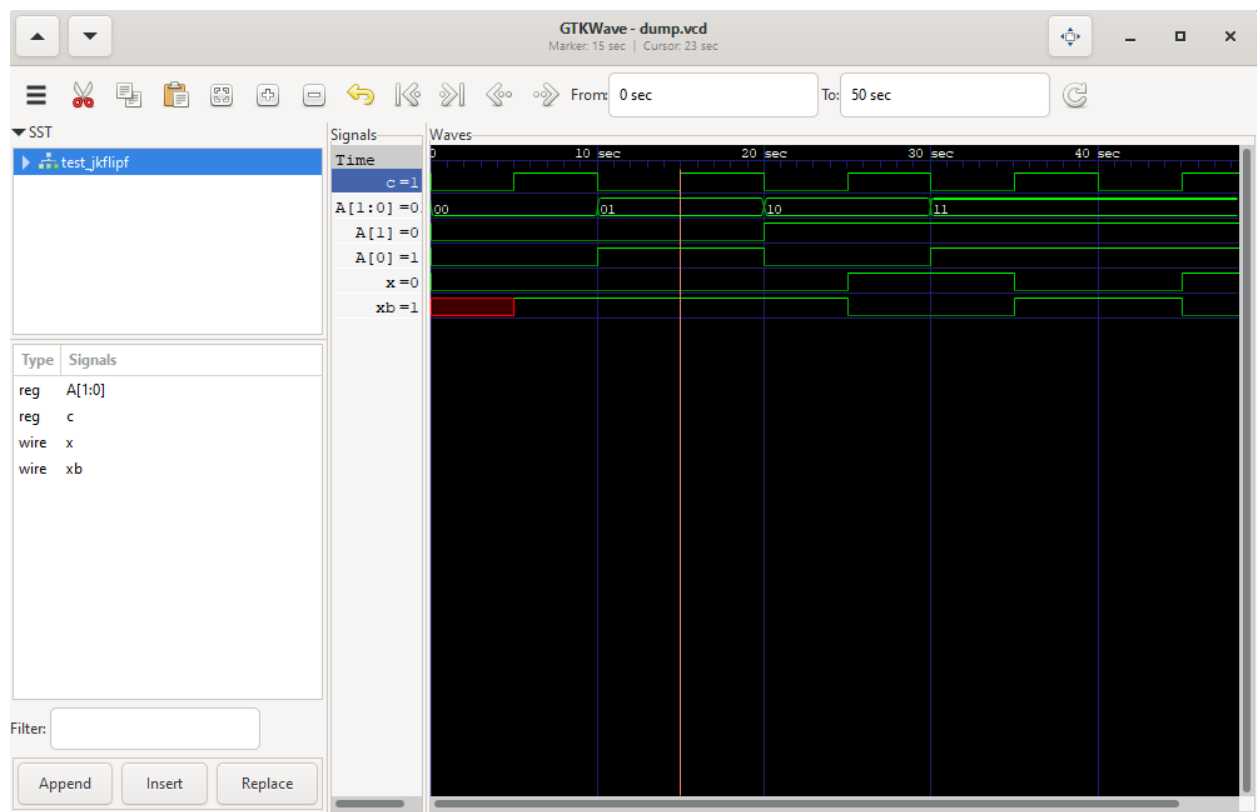
```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\iverilog\AAT\jk-ff>iverilog -o out jk-ff.v
C:\iverilog\AAT\jk-ff>vvp out
VCD info: dumpfile dump.vcd opened for output.
jk-ff.v:34: $finish called at 50 (1s)

C:\iverilog\AAT\jk-ff>gtkwave dump.vcd

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

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



Experiment 9

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.

MAIN MODULE

```
module Rshiftregister(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
```

TESTBENCH MODULE

```
module test_Rshiftregister;  
  
    reg clk, clrb, SDR;  
  
    wire[3:0]Q;  
  
    Rshiftregister RS(clk, clrb , SDR , Q);  
  
    initial clk=0;  
  
    always #50 clk=~clk;  
  
    initial  
  
    begin  
  
        $dumpfile("dump.vcd");  
  
        $dumpvars(0 , test_Rshiftregister);  
  
        clk=1;  
  
        clrb=0;
```

```
SDR=1;

#100;

clrb=1;

SDR=1;

#150;

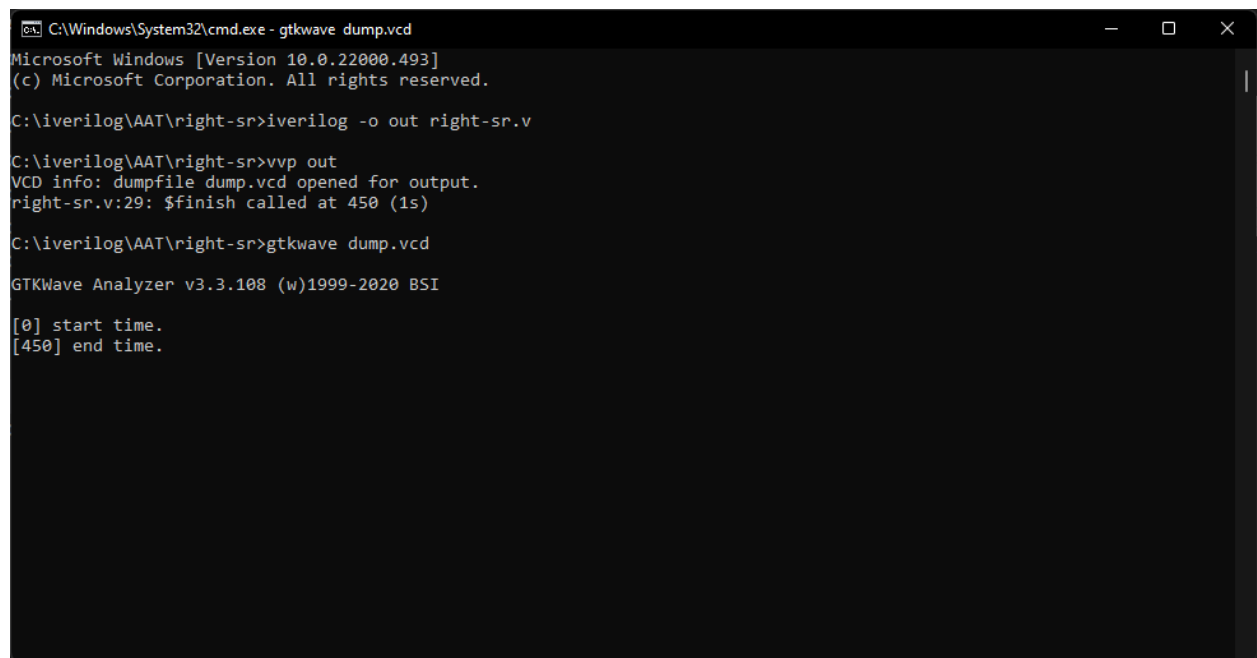
SDR=0;

#200

$finish;

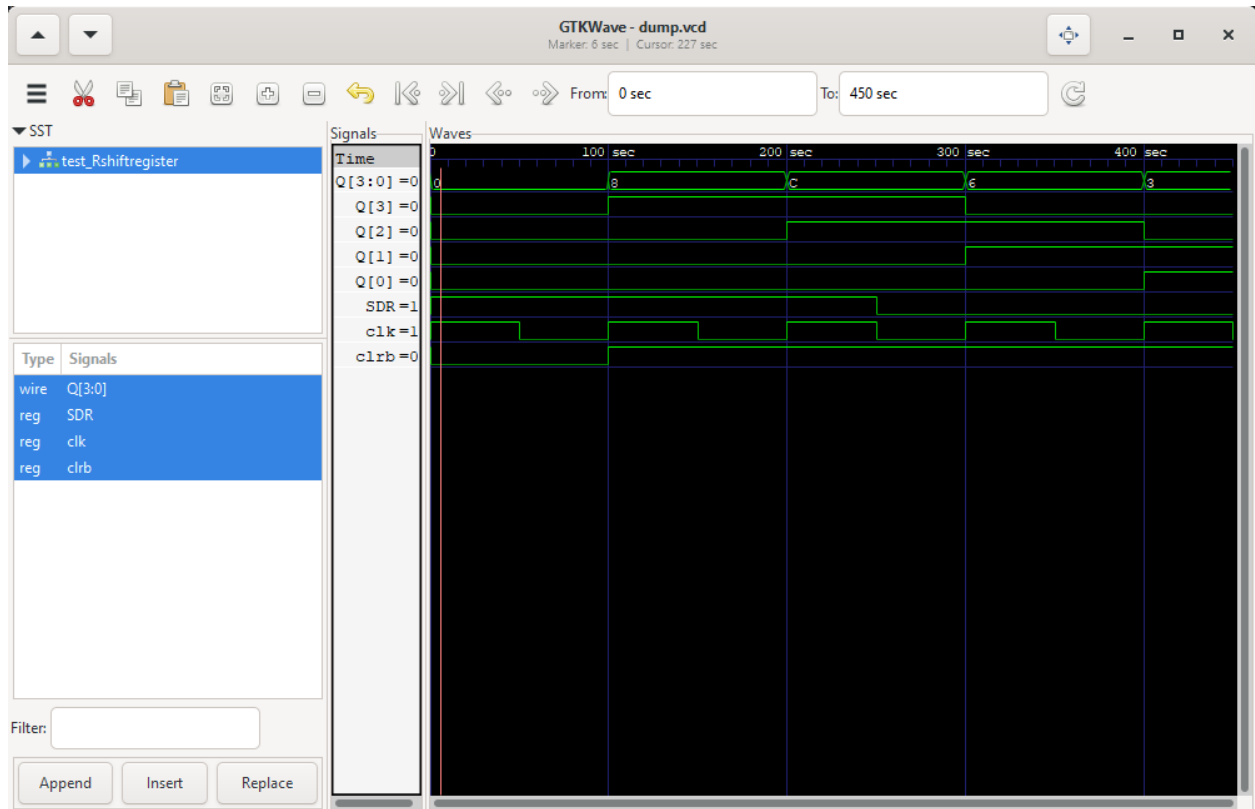
end

endmodule
```



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\iverilog\AAT\righ-sr>iverilog -o out right-sr.v
C:\iverilog\AAT\righ-sr>vvp out
VCD info: dumpfile dump.vcd opened for output.
right-sr.v:29: $finish called at 450 (1s)
C:\iverilog\AAT\righ-sr>gtkwave dump.vcd
GTKWave Analyzer v3.3.108 (w)1999-2020 BSI
[0] start time.
[450] end time.
```



Experiment 10

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

MAIN MODULE

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

TESTBENCH MODULE

```
module testbench;  
    reg r, c;  
    wire [2:0] ct;  
    counter_behav countbeh(ct,r,c);  
    initial begin  
        $dumpfile("counter_3bit.vcd");  
        $dumpvars(0,testbench);  
        r=1;  
        c=8;  
        #100;
```

```
    r=0;

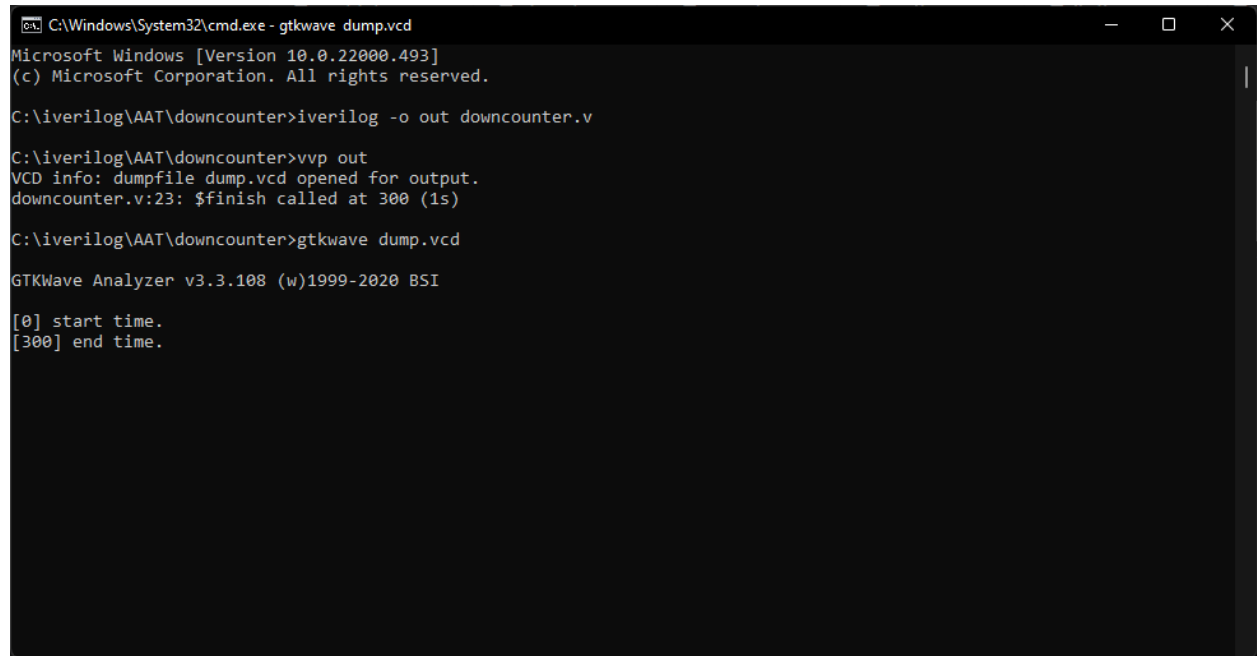
    #200

    $finish;

end

always #5 c=~c;

endmodule
```



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

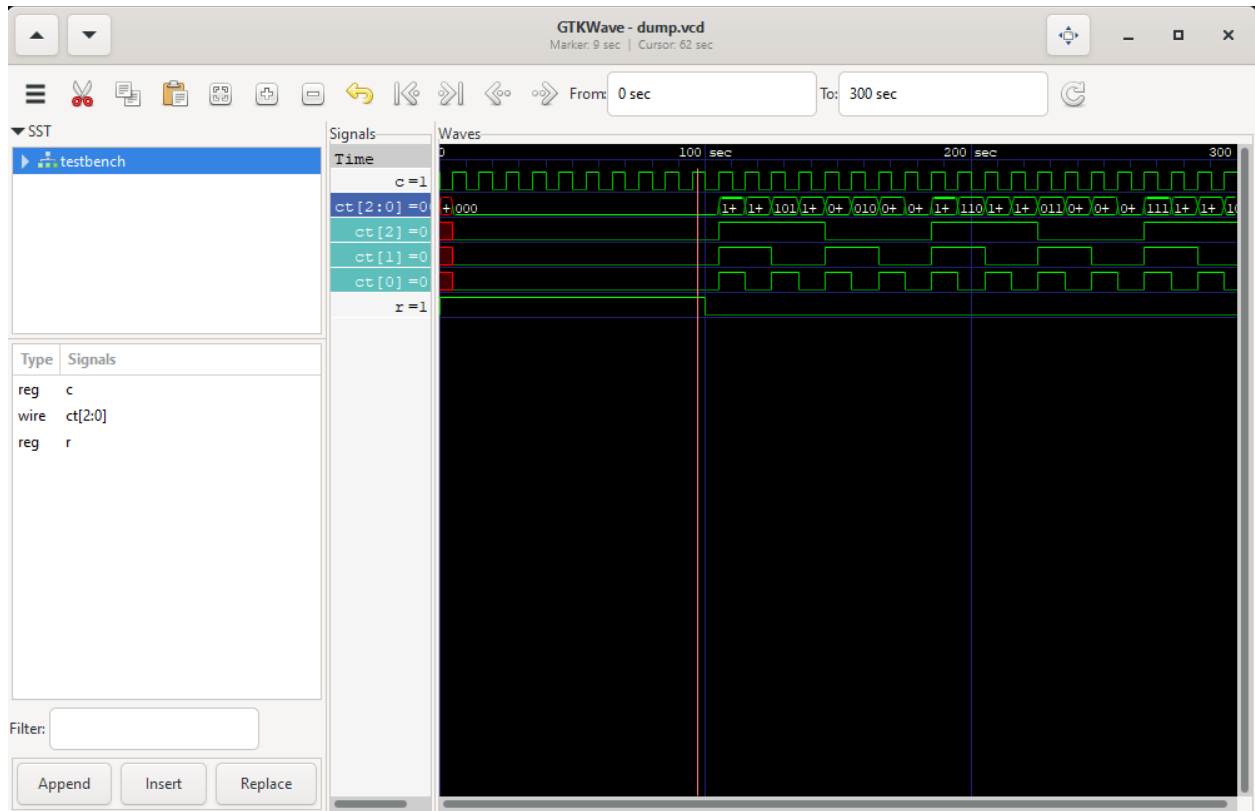
C:\iverilog\AAT\downcounter>iverilog -o out downcounter.v

C:\iverilog\AAT\downcounter>vvp out
VCD info: dumpfile dump.vcd opened for output.
downcounter.v:23: $finish called at 300 (1s)

C:\iverilog\AAT\downcounter>gtkwave dump.vcd

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

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

CYCLE III Dataflow Modeling

Experiment 11

Write HDL implementation for NAND/NOR/EXOR gates using data flow model. Simulate the same using Dataflow model and depict the timing diagram for valid inputs

MAIN MODULE

```
module NAND_NOR_EXOR (output nand1,nor1,exor1, input A, B);  
    assign nand1 = ~(A & B);  
    assign nor1 = ~(A+B);  
    assign exor1 = A ^ B;  
Endmodule
```

TESTBENCH MODULE

```
module NAND_NOR_EXOR_tb;  
    reg A, B;  
    wire nand1,nor1,exor1;  
    NAND_NOR_EXOR Indtance (nand1,nor1,exor1, A, B);  
    initial begin  
        $dumpfile("dump.vcd");  
        $dumpvars();  
        A = 1'b0; B = 1'b0;  
        #1 A = 1'b0; B = 1'b1;
```

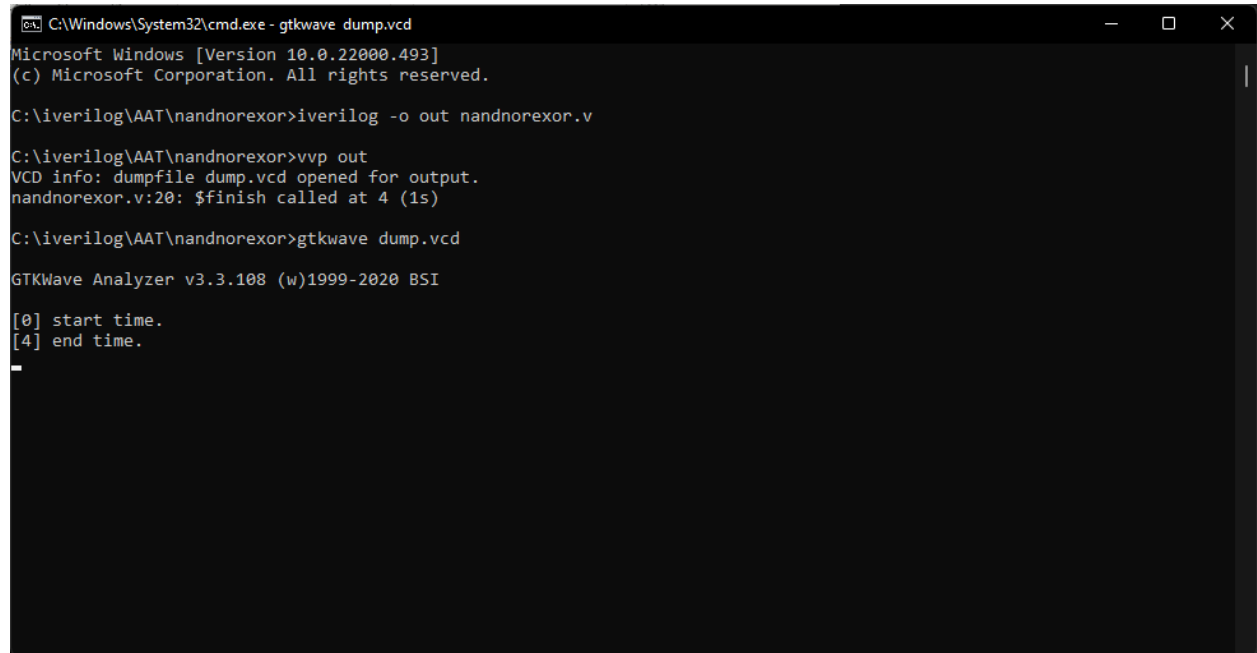
```
#1 A = 1'b1; B = 1'b0;
```

```
#1 A = 1'b1; B = 1'b1;
```

```
#1 $finish;
```

```
end
```

```
endmodule
```



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

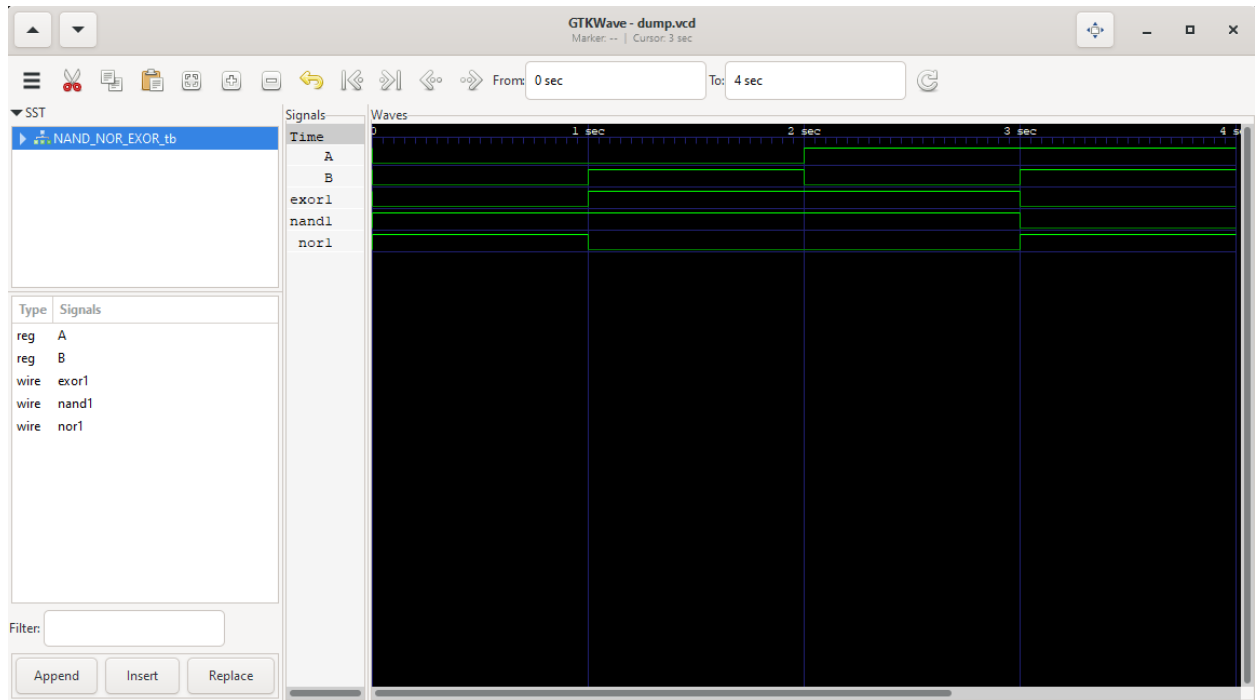
C:\iverilog\AAT\nandnorexor>iverilog -o out nandnorexor.v

C:\iverilog\AAT\nandnorexor>vvp out
VCD info: dumpfile dump.vcd opened for output.
nandnorexor.v:20: $finish called at 4 (1s)

C:\iverilog\AAT\nandnorexor>gtkwave dump.vcd

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

[0] start time.
[4] end time.
-
```



Experiment 12

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

MAIN MODULE

```
module adder(input a , input b , input cin , output s , output cout);
```

```
    assign s=a^b^cin;
```

```
    assign cout=(a&b)|(a&cin)|(b&cin);
```

```
endmodule
```

TESTBENCH MODULE

```
module test;
```

```
    reg a , b , cin;
```

```
    wire s , cout;
```

```
    adder FA( a , b ,cin , s , cout);
```

```
    initial
```

```
    begin
```

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

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

```
        a=1;
```

```
        b=1;
```

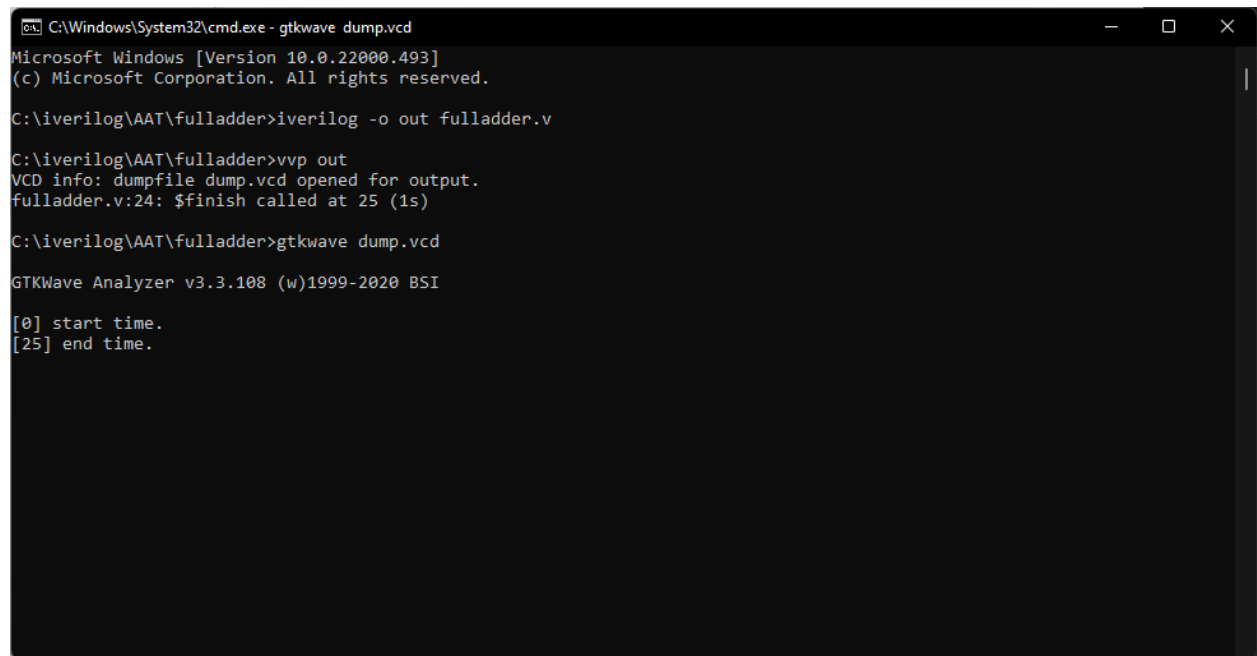
```
        cin =0 ;#5
```

```
        a=1;
```

```
        b=1;
```

```
        cin =1;#5
```

```
a=0;
b=1;
cin=0;#5
#10 $finish;
end
endmodule
```



```
C:\Windows\System32\cmd.exe - gtkwave dump.vcd
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\iverilog\AAT\fulladder>iverilog -o out fulladder.v
C:\iverilog\AAT\fulladder>vvp out
VCD info: dumpfile dump.vcd opened for output.
fulladder.v:24: $finish called at 25 (1s)
C:\iverilog\AAT\fulladder>gtkwave dump.vcd
GTKWave Analyzer v3.3.108 (w)1999-2020 BSI
[0] start time.
[25] end time.
```

