SURESH ANGADI EDUCATION FOUNDATION'S
ANGADI INSTITUTE OF TECHNOLOGY AND MANAGEMENT
Savagaon Road, BELAGAVI – 590 009.
(Approved by AICTE, New Delhi &Affiliated to Visvesvaraya Technological University, Belagavi, Accredited by NAAC)

Department of Artificial Intelligence and Data Science

Digital Design and Computer Organization
BCS302

Prepared By:

Prof. Chetan S. Patil
Assistant Professor, AI and DS, AITM, Belgaum.

Prof. Dattatreya M. Choudhari
Assistant Professor, AI and DS, AITM, Belgaum.

Syllabus

Digital Design and Computer Organization	Semester 3
Course Code BCS302	CIE Marks 50
Teaching Hours/Week (L:T:P: S) 3:0:2:0	SEE Marks 50
Total Hours of Pedagogy 40 hours Theory + 20 Hours of Practicals	Total Marks 100
Credits 04	Exam Hours 3

Examination nature (SEE) Theory

PRACTICAL COMPONENT OF IPCC

1. Given a 4-variable logic expression, simplify it using appropriate technique and simulate the same using basic gates.

2. Design a 4 bit full adder and subtractor and simulate the same using basic gates.

3. Design Verilog HDL to implement simple circuits using structural, Data flow and Behavioural model.

4. Design Verilog HDL to implement Binary Adder-Subtractor – Half and Full Adder, Half and Full Subtractor.

5. Design Verilog HDL to implement Decimal adder.

6. Design Verilog program to implement Different types of multiplexer like 2:1, 4:1 and 8:1.

7. Design Verilog program to implement types of De-Multiplexer.

8. Design Verilog program for implementing various types of Flip-Flops such as SR, JK and D.

Course outcomes (Course Skill Set):

At the end of the course, the student will be able to:
CO1: Apply the K–Map techniques to simplify various Boolean expressions.
CO2: Design different types of combinational and sequential circuits along with Verilog programs.
CO3: Describe the fundamentals of machine instructions, addressing modes and Processor performance.
CO4: Explain the approaches involved in achieving communication between processor and I/O devices.
CO5:Analyze internal Organization of Memory and Impact of cache/Pipelining on Processor Performance.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

CIE for the theory component of the IPCC (maximum marks 50)

- IPCC means practical portion integrated with the theory of the course.
- CIE marks for the theory component are 25 marks and that for the practical component is 25 marks.
- 25 marks for the theory component are split into 15 marks for two Internal Assessment Tests (Two Tests, each of 15 Marks with 01-hour duration, are to be conducted) and 10 marks for other assessment methods mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.
- Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for 25 marks).
- The student has to secure 40% of 25 marks to qualify in the CIE of the theory component of IPCC.

CIE for the practical component of the IPCC

- 15 marks for the conduction of the experiment and preparation of laboratory record, and 10 marks for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.
- The laboratory test (duration 02/03 hours) after completion of all the experiments shall be conducted for 50 marks and scaled down to 10 marks.
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for 25 marks.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

SEE for IPCC

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the course (duration 03 hours)

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), should have a mix of topics under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.
4. Marks scored by the student shall be proportionally scaled down to 50 Marks

The theory portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper may include questions from the practical component.

CONTENTS

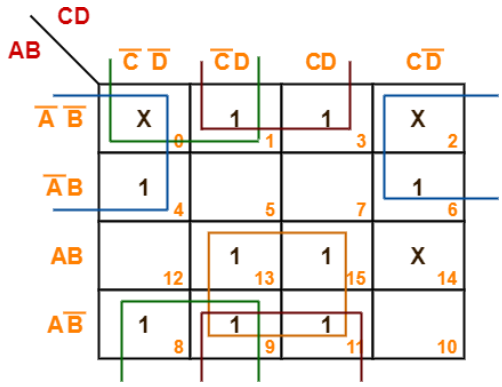
1. Given a 4-variable logic expression, simplify it using appropriate technique and simulate the same using basic gates.

We can minimize Boolean expressions of 3, 4 variables very easily using K-map without using any Boolean algebra theorems. K-map can take two forms Sum of Product (SOP) and Product of Sum (POS) according to the need of problem. K-map is table like representation but it gives more information than TRUTH TABLE. We fill grid of K-map with 0's and 1's then solves it by making groups.

Components Required:

1. NOT Gate 7404—2 Nos
2. AND Gate 7408 ---2 Nos
3. OR Gate 7432--- 2 Nos
4. Digital trainer kit
5. Patch Chords

$F(ABCD)= \sum m\ (1,3,4,6,8,9,11,13,15) + d\ (0,2,14)$



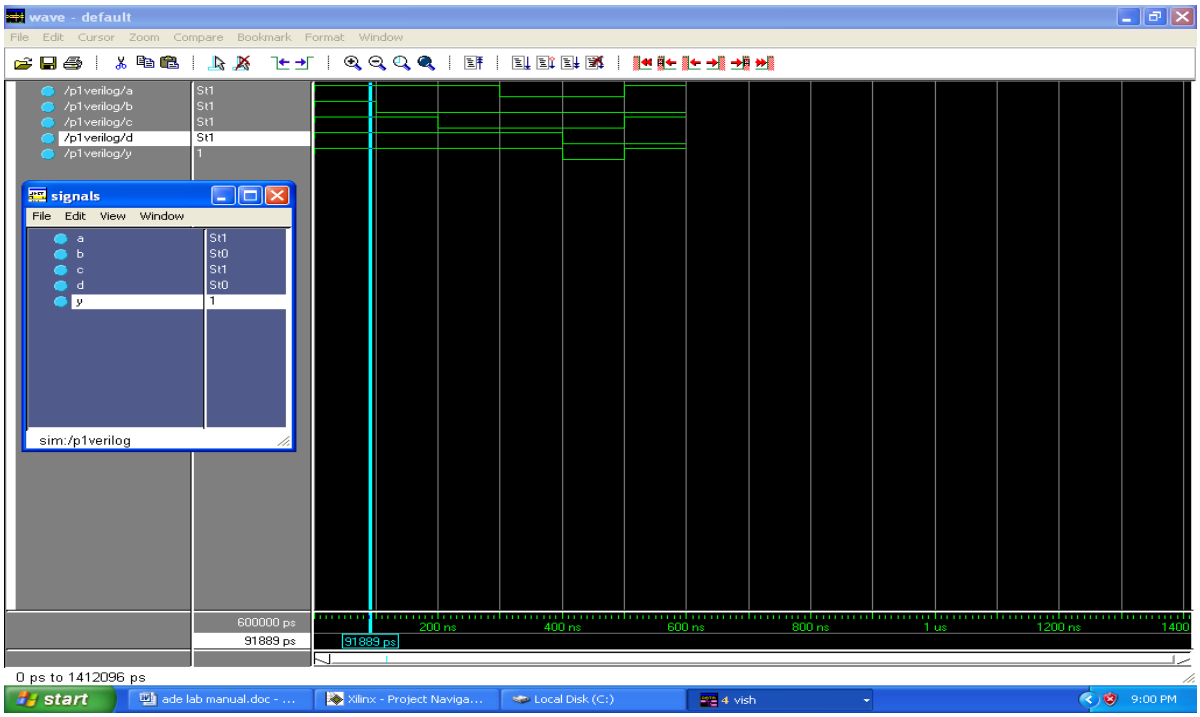
F(ABCD)= A'D' + B'C'+B'D+AD OR F(ABCD)= A'D' + B'C'+A'B'+AD

Truth Table to be Verified: -

SL NO	Decimal Value	MINTERMS	m terms	A	B	C	D	A'	B'	C'	D'	A'D'	B'C'	B'D	AD	F(ABCD)=A'D'+B'C'+B'D+AD
1	0	A'B'C'D'	m0	0	0	0	0	1	1	1	1	1	1	1	0	1
2	1	A'B'C'D	m1	0	0	0	1	1	1	1	0	0	1	0	0	1
3	2	A'B'CD'	m2	0	0	1	0	1	1	0	1	1	0	1	0	1
4	3	A'B'CD	m3	0	0	1	1	1	1	0	0	0	0	1	0	1
5	4	A'BC'D'	m4	0	1	0	0	1	0	1	1	1	0	0	0	1
6	5	A'BC'D	m5	0	1	0	1	1	0	1	0	0	0	0	0	0
7	6	A'BCD'	m6	0	1	1	0	1	0	0	1	1	0	0	0	1
8	7	A'BCD	m7	0	1	1	1	1	0	0	0	0	0	0	0	0
9	8	A'B'C'D'	m8	1	0	0	0	0	1	1	1	0	1	1	0	1
10	9	AB'C'D	m9	1	0	0	1	0	1	1	0	0	1	0	1	1
11	10	AB'CD'	m10	1	0	1	0	0	1	0	1	0	0	0	0	0
12	11	AB'CD	m11	1	0	1	1	0	1	0	0	0	0	0	1	1
13	12	ABC'D'	m12	1	1	0	0	0	0	1	1	0	0	0	0	0
14	13	ABC'D	m13	1	1	0	1	0	0	1	0	0	0	0	1	1
15	14	ABCD'	m14	1	1	1	0	0	0	0	1	0	0	0	0	0
16	15	ABCD	m15	1	1	1	1	0	0	0	0	0	0	0	1	1

```
module p1verilog(a,b,c,d,y);
   input a;
   input   b;
   input   c;
   input   d;
   output y;

        reg y;
        always @ (a, b,c,d)
        begin
        y= (~a & ~d) | (~b & ~c) | (~b & d) | (a & d);
        end
        endmodule
```

OUTPUT:

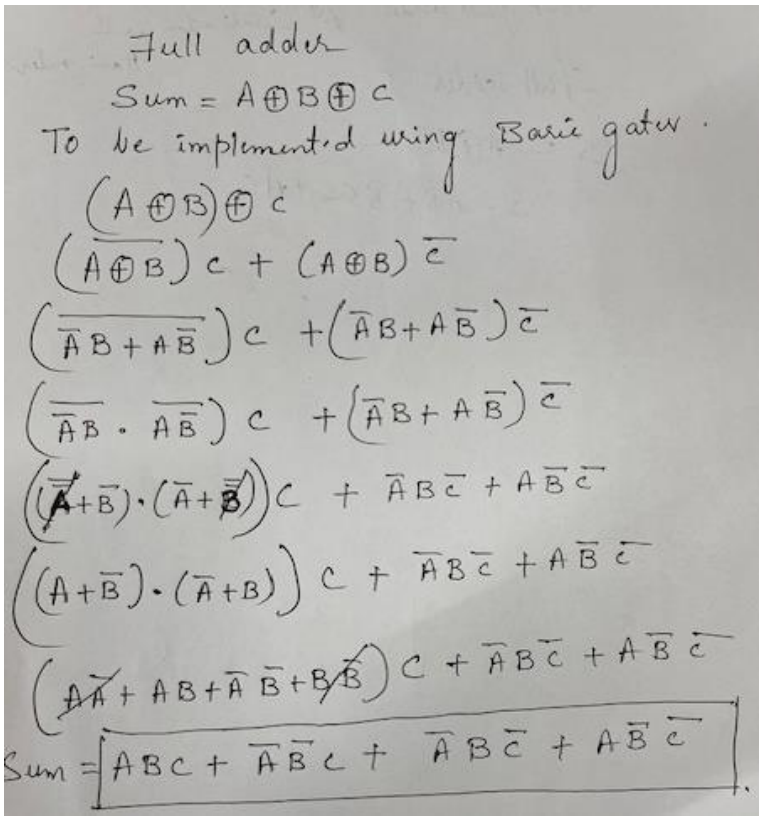


2. Design a 4-bit full adder and subtractor and simulate the same using basic gates.

Full adder

BOOLEAN EXPRESSIONS:

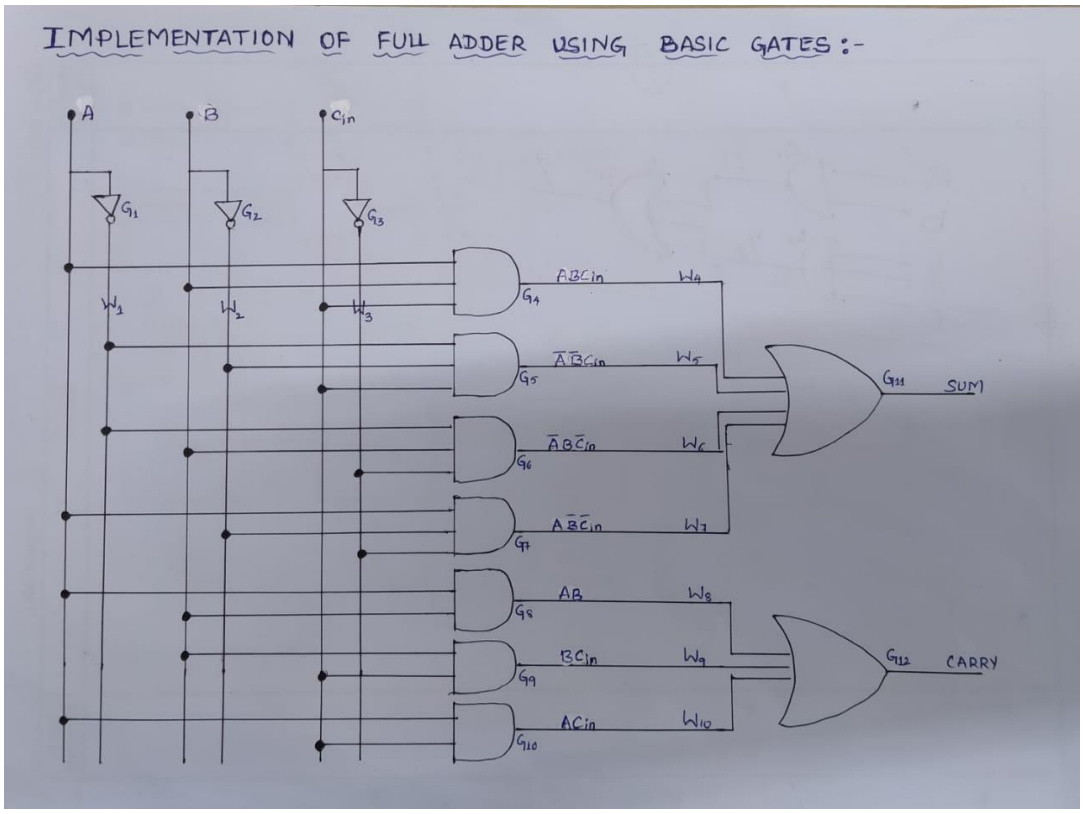
sum= $A \oplus B \oplus C$

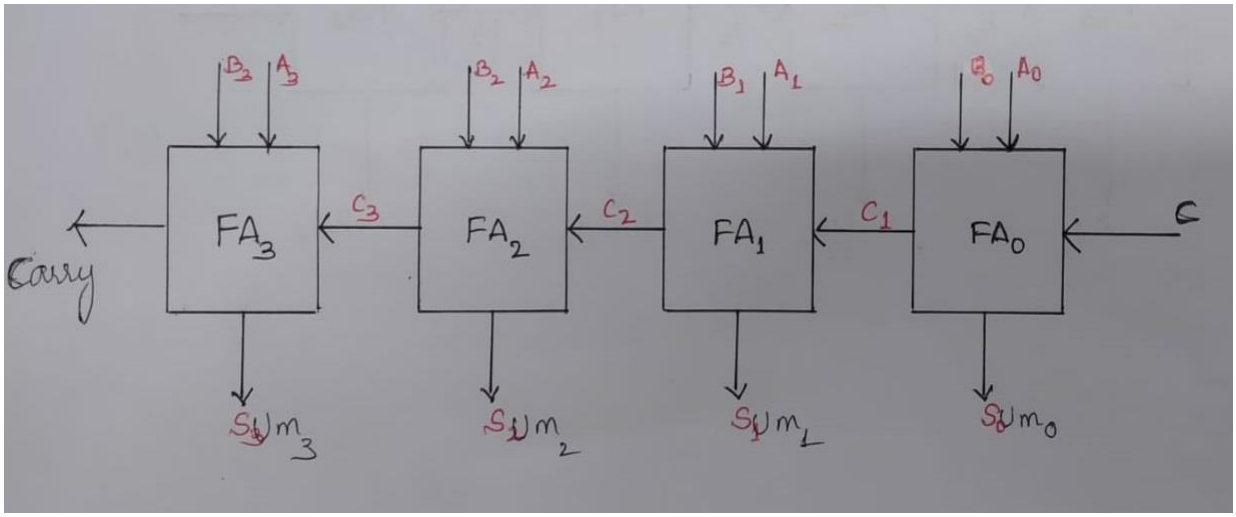


carry=A B + B C + A C

TRUTH TABLE:

INPUTS			OUTPUTS	
A	**B**	**C**	**sum**	**carry**
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



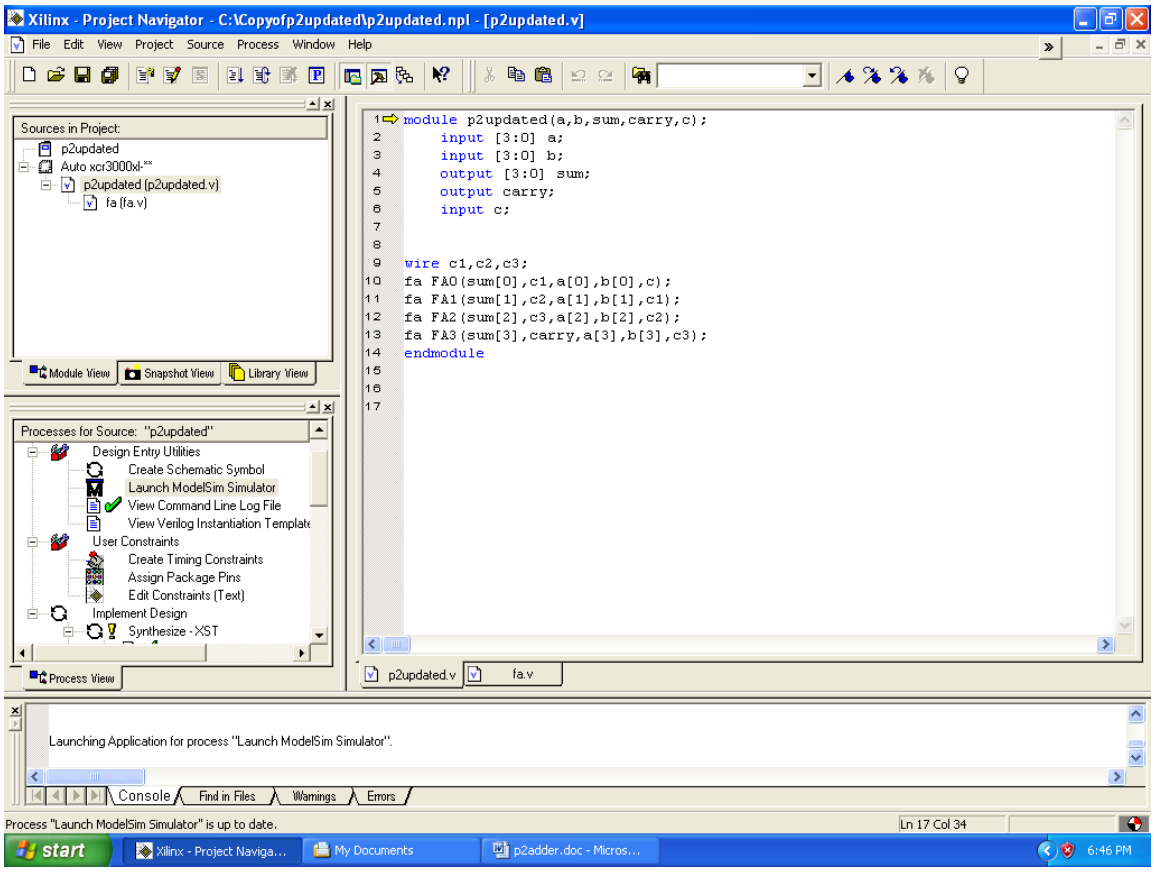


Full adder: -

Step 1

```
module p2updated(a,b,sum,carry,c);
   input [3:0] a;
   input [3:0] b;
   output [3:0] sum;
   output carry;
```

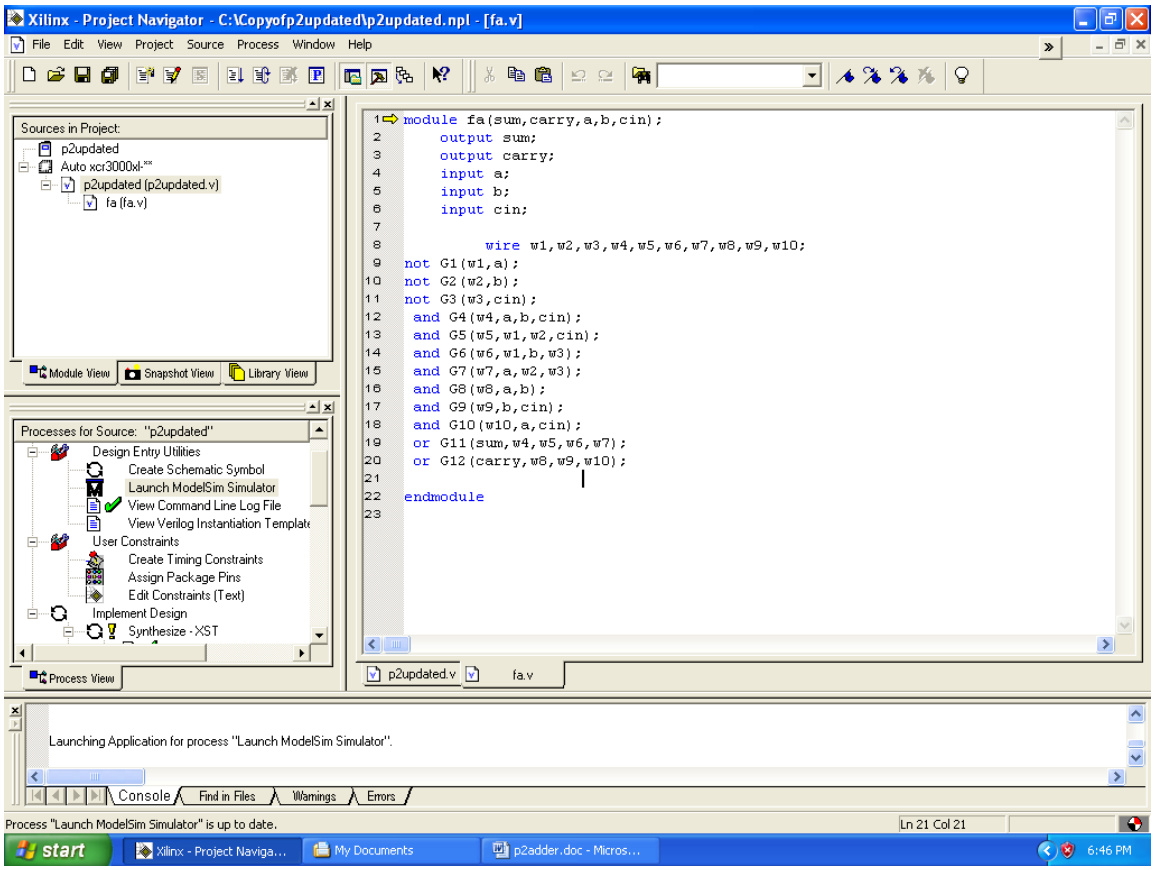
```
   input c;
 wire c1,c2,c3;
fa FA0(sum[0],c1,a[0],b[0],c);
fa FA1(sum[1],c2,a[1],b[1],c1);
fa FA2(sum[2],c3,a[2],b[2],c2);
fa FA3(sum[3],carry,a[3],b[3],c3);
endmodule
```



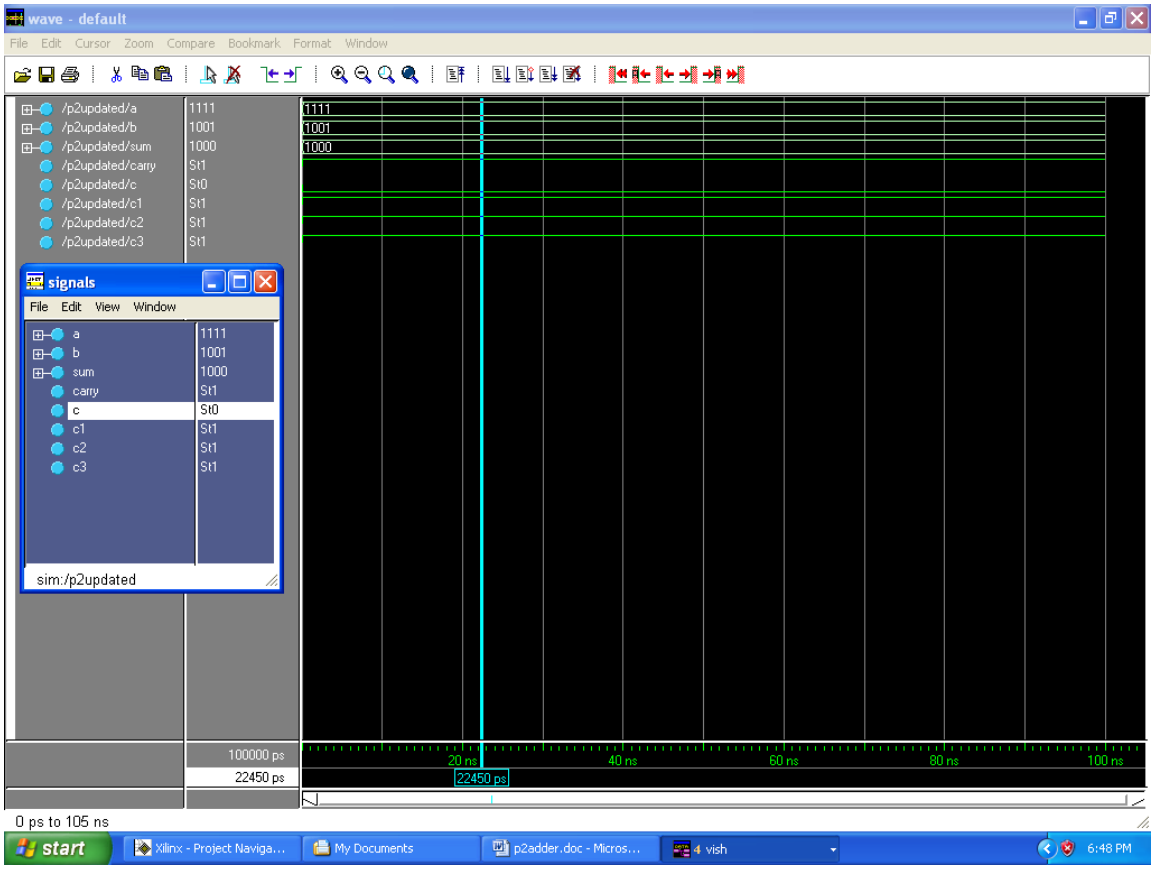
Step 2 (create the file fa.v under the main module p2updated.v)

i)right click on main module p2updated.v
ii) select new source ->Verilog module ->enter file name as fa.v
iii) fa.v sub module gets created under the main module as seen in the process window
iv)type the code in the sub module fa.v and save.
v)compile both the main module p2updated.v and sub module fa.v

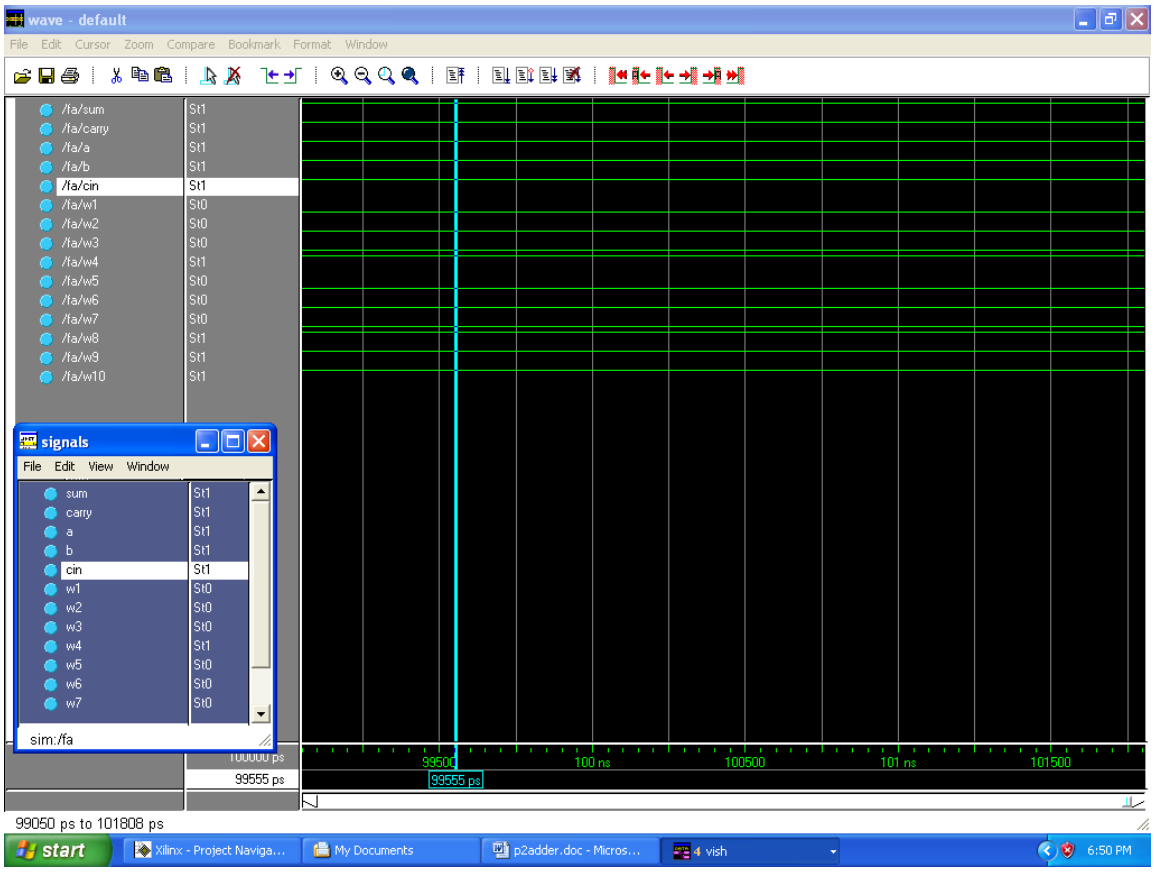
```
module fa(sum,carry,a,b,cin);
   output sum;
   output carry;
   input  a;
   input b;
   input cin;

        wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10;
 not G1(w1,a);
 not G2(w2,b);
 not G3(w3,cin);
  and  G4(w4,a,b,cin);
  and G5(w5,w1,w2,cin);
  and G6(w6,w1,b,w3);
  and G7(w7,a,w2,w3);
  and G8(w8,a,b);
  and G9(w9,b,cin);
  and G10(w10,a,cin);
  or G11(sum,w4,w5,w6,w7);
  or G12(carry,w8,w9,w10);

endmodule
```



Step1 output



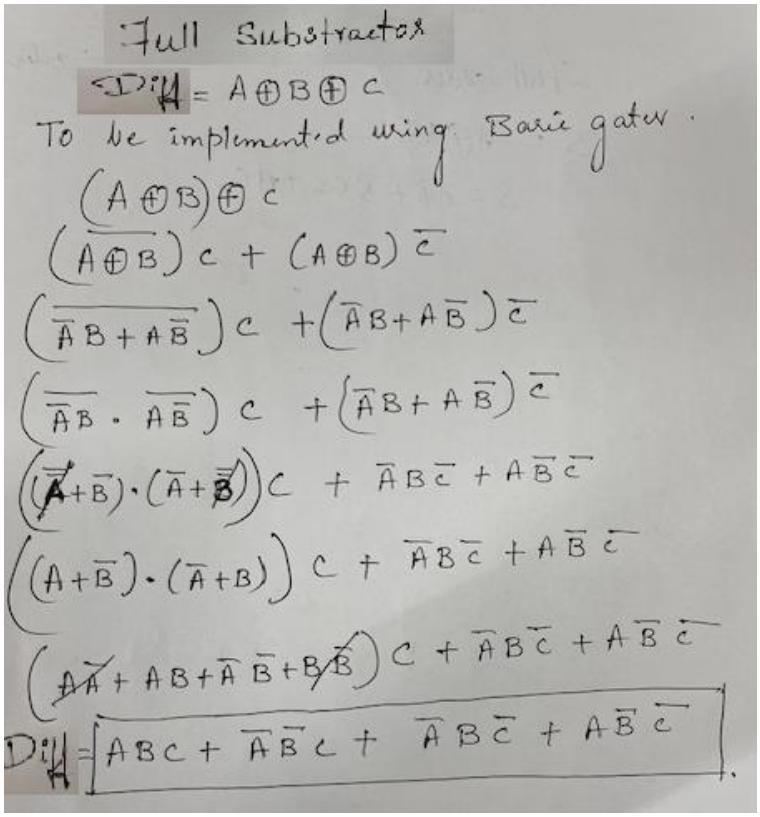
Step2 output



BOOLEAN EXPRESSIONS:

Full subtractor

diff=A⊕B⊕C

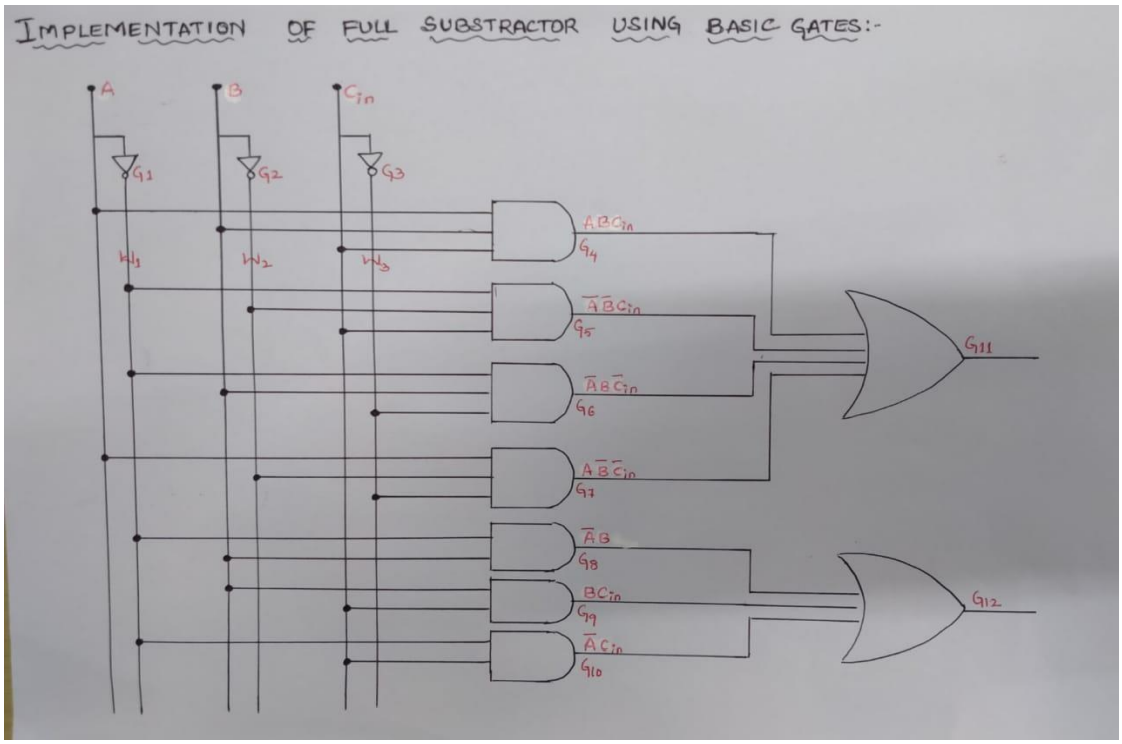


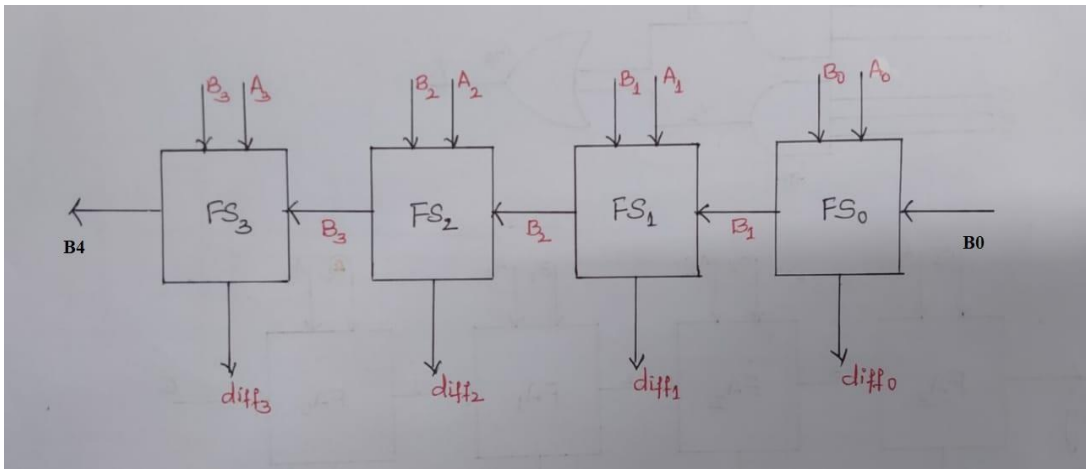
borr= $\overline{A}$ B + B C + $\overline{A}$ C

TRUTH TABLE

INPUTS			OUTPUTS	
A	B	C	diff	borr
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

Full subtractor: -

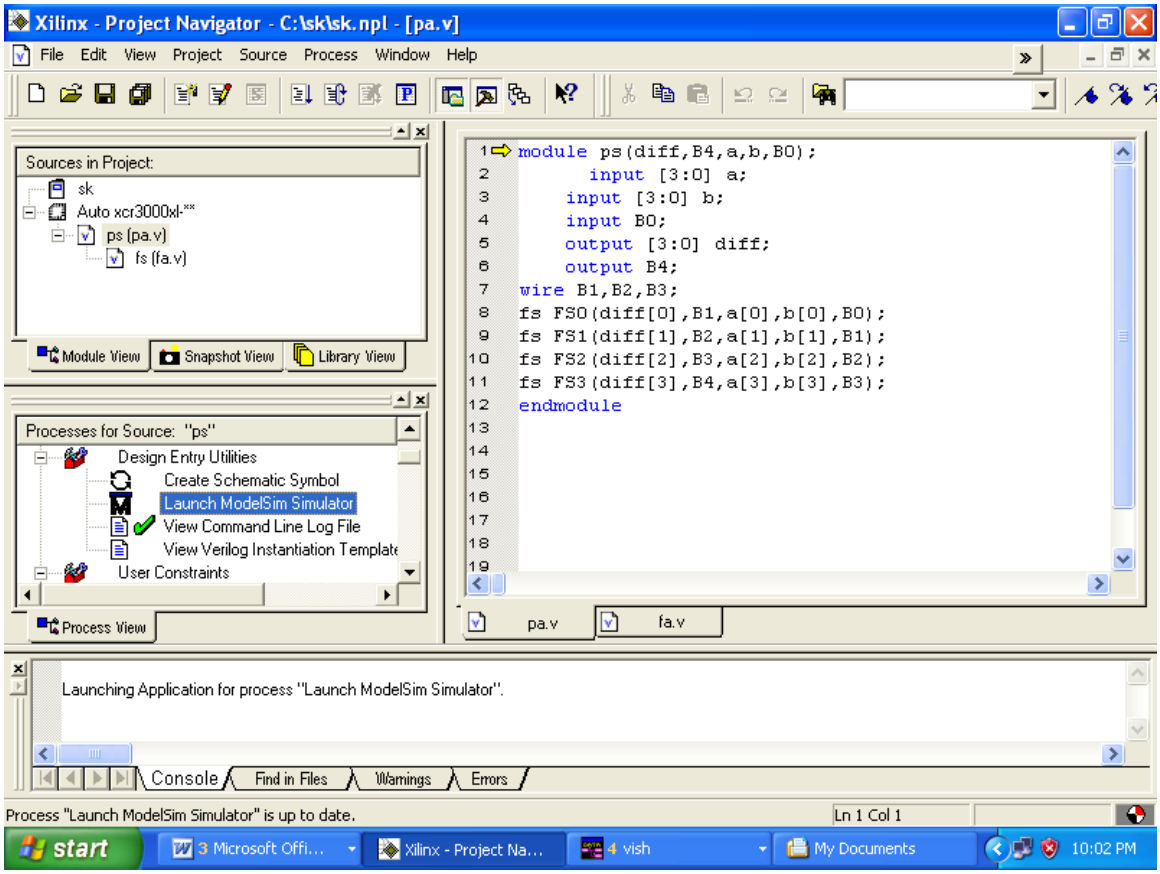




Changes to be made for full subtractor

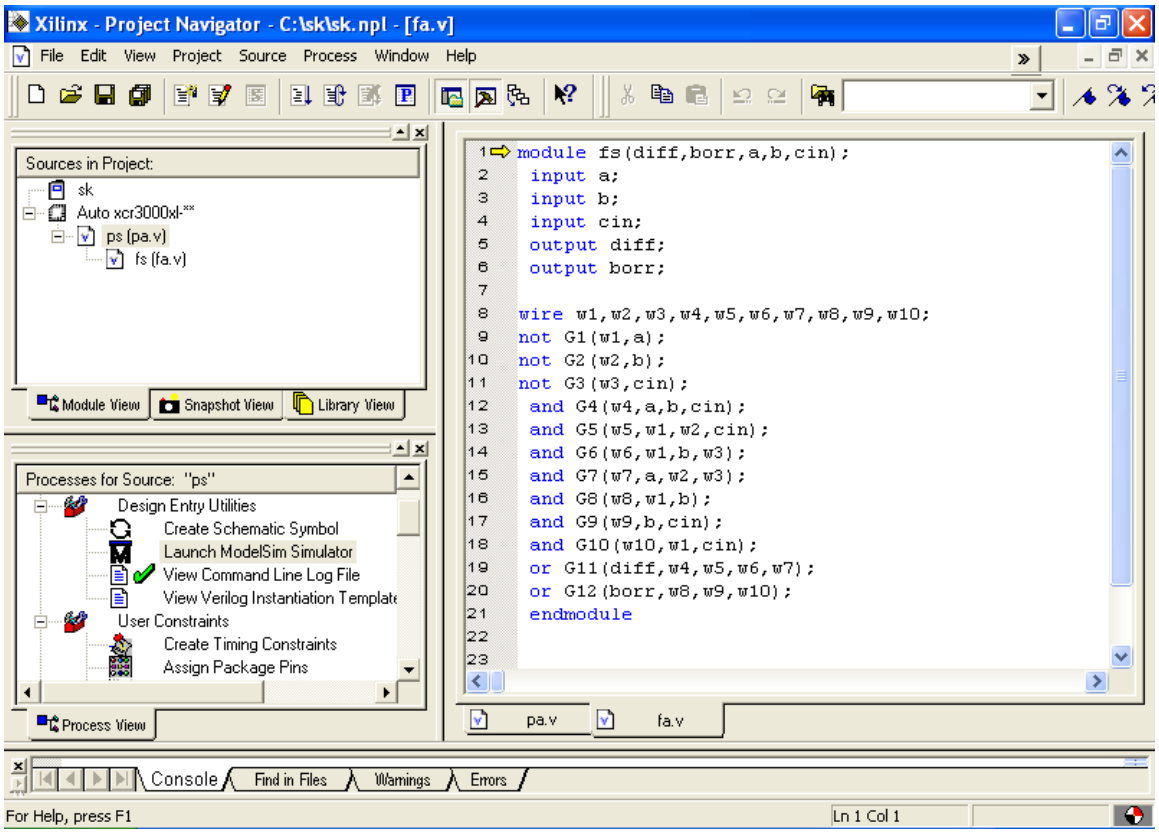
Step1

```
module ps(diff,B4,a,b,B0);
input [3:0] a;
input [3:0] b;
input B0;
output [3:0] diff;
output B4;
wire B1,B2,B3;
fs FS0(diff[0],B1,a[0],b[0],B0);
fs FS1(diff[1],B2,a[1],b[1],B1);
fs FS2(diff[2],B3,a[2],b[2],B2);
fs FS3(diff[3],B4,a[3],b[3],B3);
endmodule
```

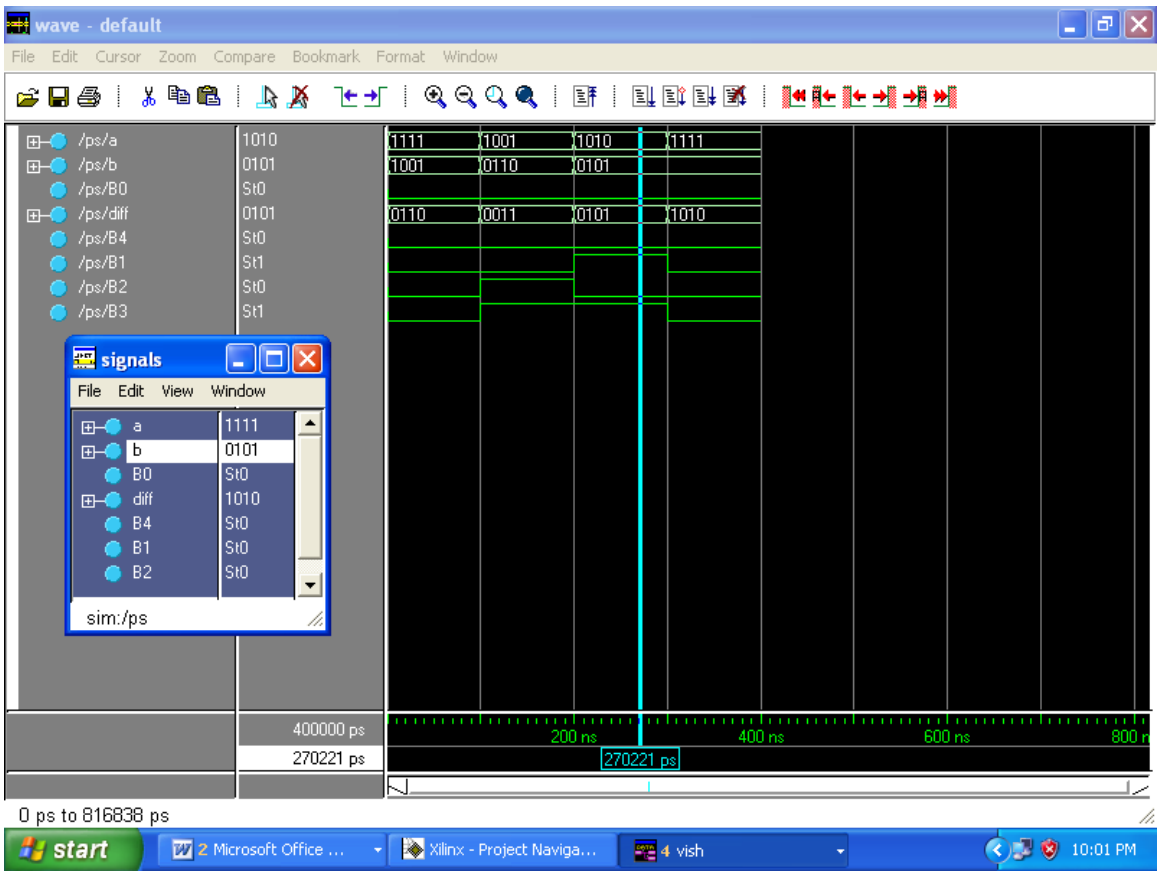


step2

```
module fs(diff,borr,a,b,cin);
 input a;
 input b;
 input cin;
 output diff;
 output borr;

wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10;
not G1(w1,a);
not G2(w2,b);
not G3(w3,cin);
 and G4(w4,a,b,cin);
 and G5(w5,w1,w2,cin);
 and G6(w6,w1,b,w3);
 and G7(w7,a,w2,w3);
 and G8(w8,w1,b);
 and G9(w9,b,cin);
 and G10(w10,w1,cin);
 or G11(diff,w4,w5,w6,w7);
 or G12(borr,w8,w9,w10);
 endmodule
```


Xilinx - Project Navigator - C:\sk\sk.npl - [fa.v]
File Edit View Project Source Process Window Help
Sources in Project:
sk
Auto xcr3000xl-**
ps (pa.v)
fs (fa.v)
Module View
Snapshot View
Library View
Processes for Source: "ps"
Design Entry Utilities
Create Schematic Symbol
Launch ModelSim Simulator
View Command Line Log File
View Verilog Instantiation Templat
User Constraints
Create Timing Constraints
Assign Package Pins
Process View
1 module fs(diff,borr,a,b,cin);
2 input a;
3 input b;
4 input cin;
5 output diff;
6 output borr;
7
8 wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10;
9 not G1(w1,a);
10 not G2(w2,b);
11 not G3(w3,cin);
12 and G4(w4,a,b,cin);
13 and G5(w5,w1,w2,cin);
14 and G6(w6,w1,b,w3);
15 and G7(w7,a,w2,w3);
16 and G8(w8,w1,b);
17 and G9(w9,b,cin);
18 and G10(w10,w1,cin);
19 or G11(diff,w4,w5,w6,w7);
20 or G12(borr,w8,w9,w10);
21 endmodule
22
23
pa.v
fa.v
Console
Find in Files
Warnings
Errors
For Help, press F1
Ln 1 Col 1

Output:-

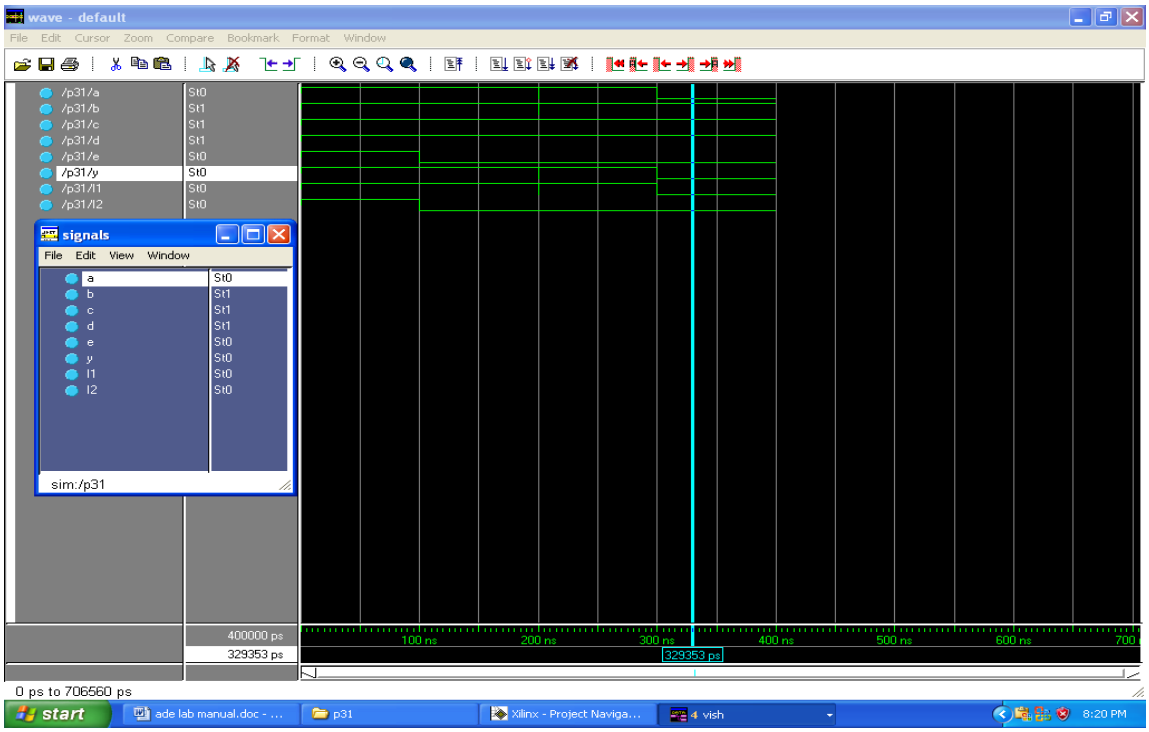

wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/ps/a
/ps/b
/ps/B0
/ps/diff
/ps/B4
/ps/B1
/ps/B2
/ps/B3
1010
0101
St0
0101
St0
St1
St0
St1
1111 1001 1010 1111
1001 0110 0101
0110 0011 0101 1010
signals
File Edit View Window
a 1111
b 0101
B0 St0
diff 1010
B4 St0
B1 St0
B2 St0
sim:/ps
400000 ps
270221 ps
200 ns
400 ns
600 ns
270221 ps
0 ps to 816838 ps
start
2 Microsoft Office ...
Xilinx - Project Naviga...
4 vish
10:01 PM

3. Design Verilog HDL to implement simple circuits using structural, Data flow and Behavioural model.

Structural Model

```
module p3structural(a,b,c,d,e,y);
   input a;
   input b;
    input c;
   input d;
    input e;
  output y;
wire Y1,Y2;
and G1(Y1,a,b);
and G2(Y2,c,d,e);
or G3(Y,Y1,Y2);
endmodule
```

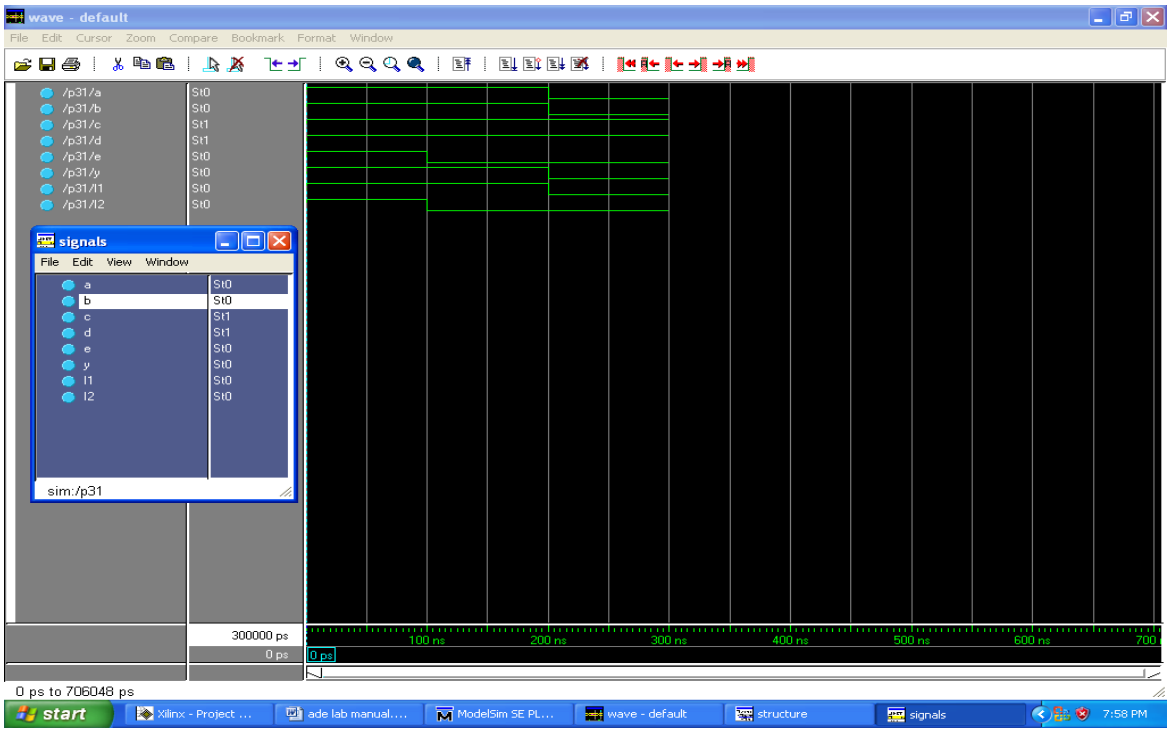
OUTPUT: -



Data Flow Model

```
module p31(a,b,c,d,e,y);
   input a;
   input b;
   input c;
   input d;
   input e;
   output y;

  wire Y1,Y2;
  assign Y1=a & b;
  assign Y2= c&d&e;
assign y= Y1|Y2;
endmodule
```

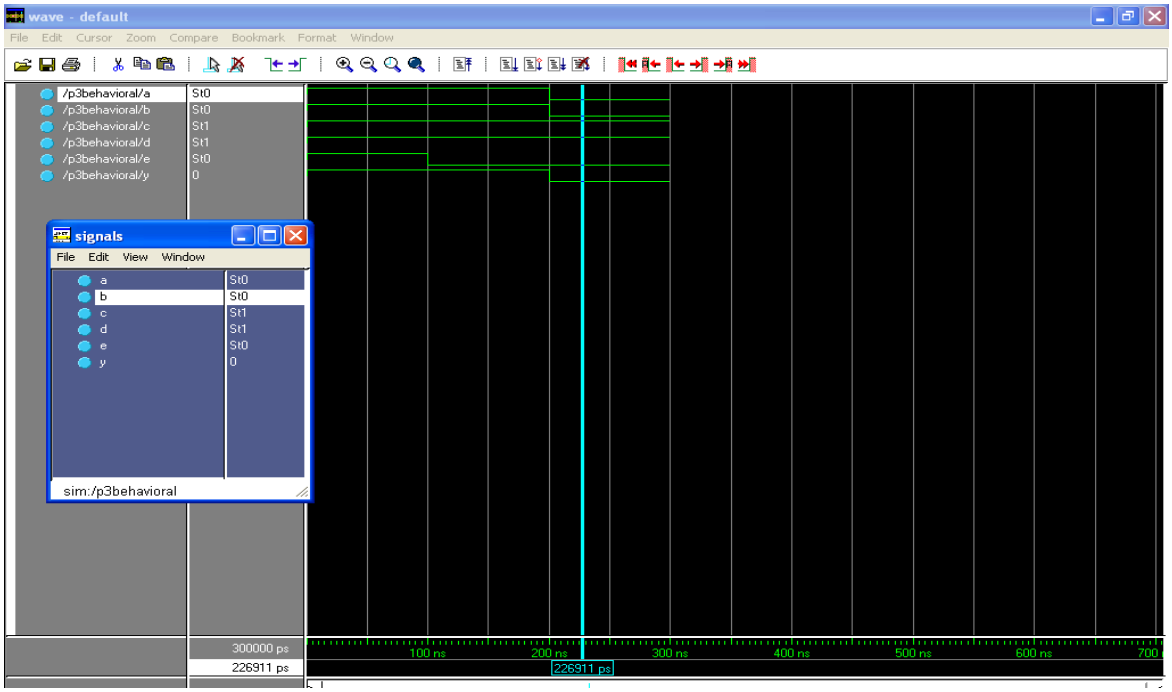
OUTPUT: -



Behavioral Model

```
module p3behavioral(a,b,c,d,e,y);
   input a;
   input b;
   input c;
   input d;
   input e;
   output y;

reg y;
always @(a,b,c,d,e)
begin
y= (a & b) | (c & d &e);
end
endmodule
```

OUTPUT: -



4.Design Verilog HDL to implement Binary Adder-Subtractor – Half and Full Adder, Half and Full Subtractor.

Half adder: -

BOOLEAN EXPRESSIONS:

sum=A ⊕ B

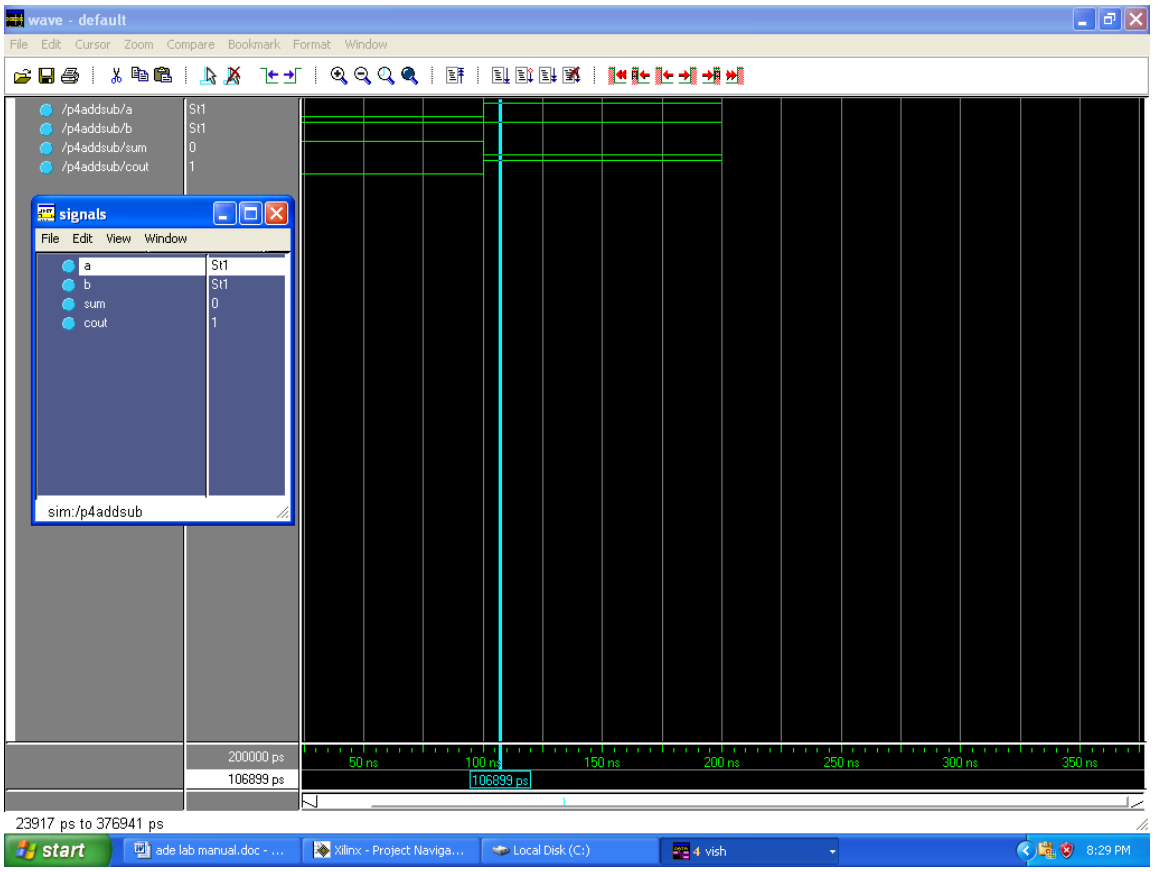
cout=A B

TRUTH TABLE

INPUTS		OUTPUTS	
A	**B**	**sum**	**cout**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

```
module p4addsub(a,b,sum,cout);
   input a;
   input b;
   output sum;
   output cout;

reg sum, cout;
always @(a,b)
begin
sum = a ^ b;
cout = a & b;
end
endmodule
```

OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p4addsub/a
/p4addsub/b
/p4addsub/sum
/p4addsub/cout
St1
St1
0
1
signals
File Edit View Window
a
b
sum
cout
St1
St1
0
1
sim:/p4addsub
200000 ps
106899 ps
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
106899 ps
23917 ps to 376941 ps
start
ade lab manual.doc - ...
Xilinx - Project Naviga...
Local Disk (C:)
4 vish
8:29 PM

Half Subtractor

BOOLEAN EXPRESSIONS:

Diff = A ⊕ B

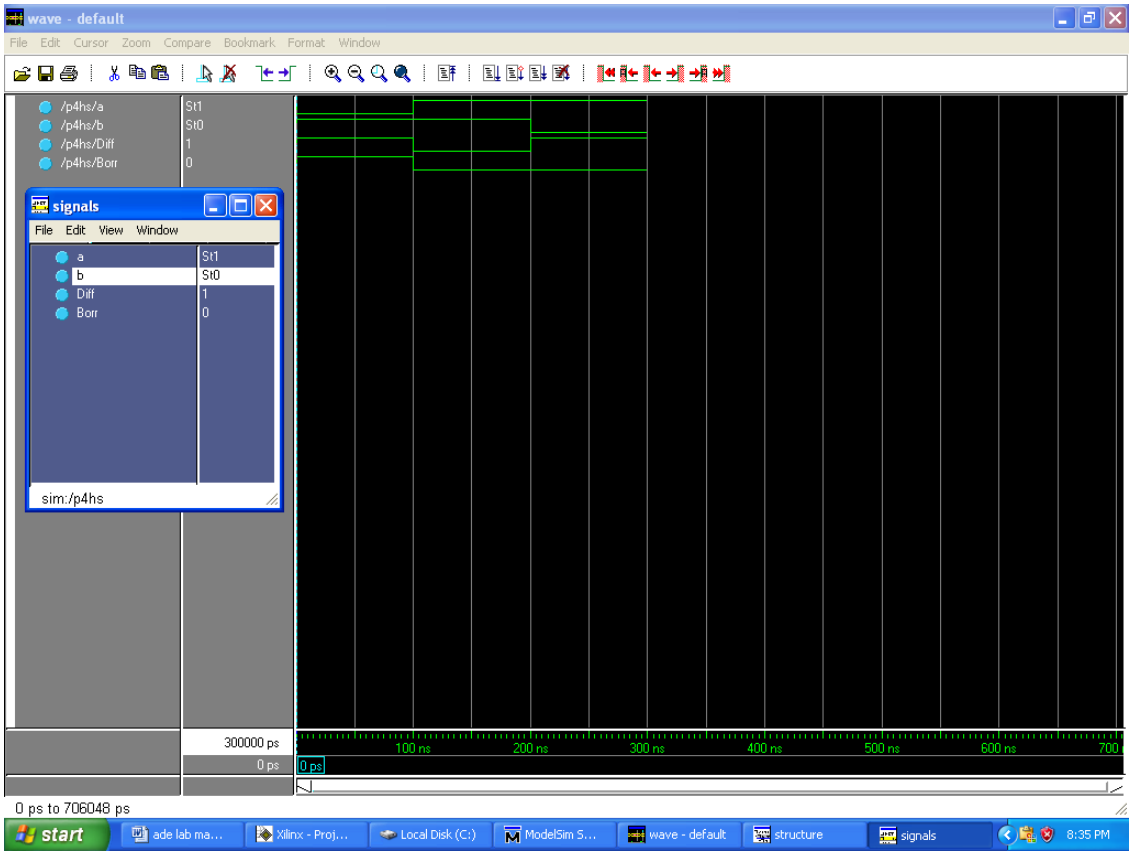
Borr = $\bar{A}$B

TRUTH TABLE

INPUTS		OUTPUTS	
A	B	Diff	Borr
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

```
module p4hs(a,b,Diff,Borr);
   input a;
   input b;
   output Diff;
   output Borr;

reg Diff, Borr;
always @(a,b)
begin
Diff = a ^ b;
Borr = (~ a) & b;
end
endmodule
```

OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p4hs/a
/p4hs/b
/p4hs/Diff
/p4hs/Borr
St1
St0
1
0
signals
File Edit View Window
a
b
Diff
Borr
St1
St0
1
0
sim:/p4hs
300000 ps
0 ps
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
0 ps
0 ps to 706048 ps
start
ade lab ma...
Xilinx - Proj...
Local Disk (C:)
ModelSim S...
wave - default
structure
signals
8:35 PM

Full Adder

BOOLEAN EXPRESSIONS

Sum= A ⊕ B ⊕ Cin

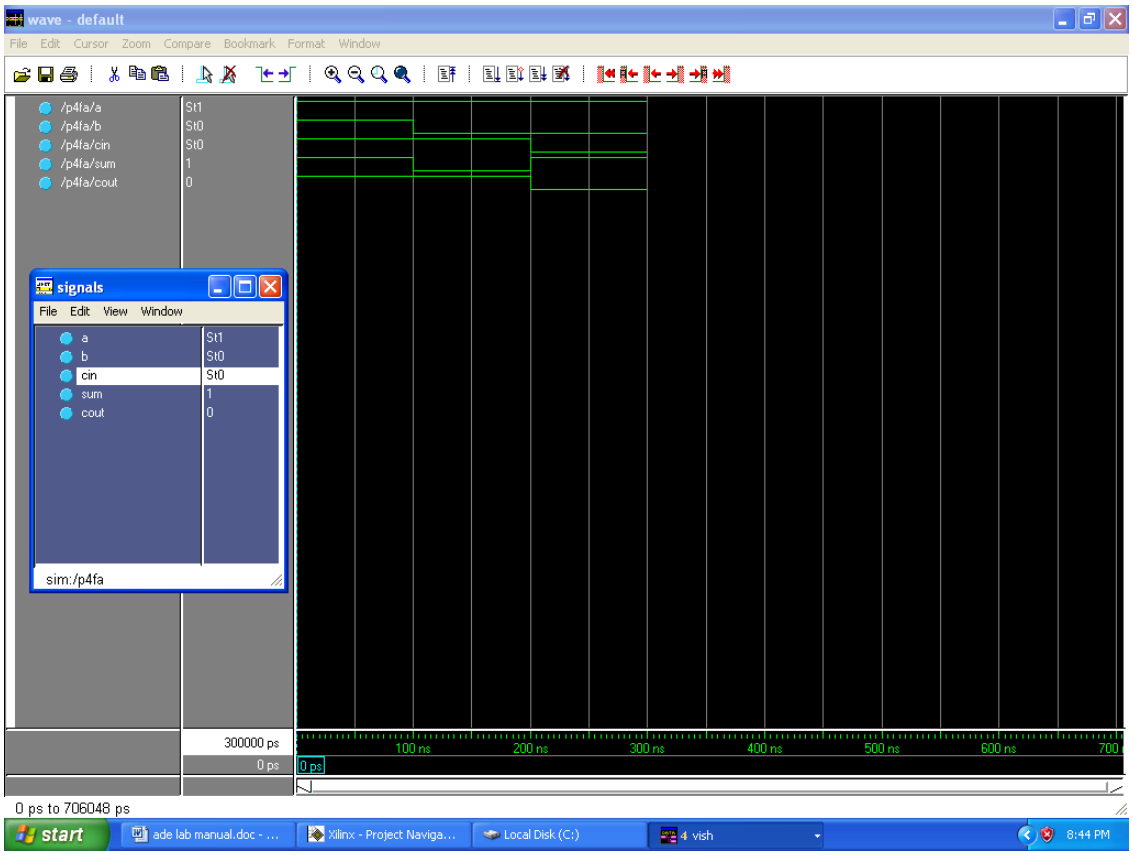
Cout=A B + B Cin + A Cin

TRUTH TABLE

INPUTS			OUTPUTS	
A	**B**	**Cin**	**sum**	**cout**
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

```
module p4fa(a,b,cin,sum,cout);
   input a;
   input b;
   input cin;
   output sum;
   output cout;
reg sum, cout;
always @(a,b,cin)
begin
sum = a^b^cin;
cout= (a & b) | (b & cin) | (a & cin) ;
end
endmodule
```

OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p4fa/a
/p4fa/b
/p4fa/cin
/p4fa/sum
/p4fa/cout
St1
St0
St0
1
0
signals
File Edit View Window
a
b
cin
sum
cout
St1
St0
St0
1
0
sim:/p4fa
300000 ps
0 ps
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
0 ps
0 ps to 706048 ps
start
ade lab manual.doc - ...
Xilinx - Project Naviga...
Local Disk (C:)
4 vish
8:44 PM

Full Subtractor

BOOLEAN EXPRESSIONS

Diff=A⊕B⊕C

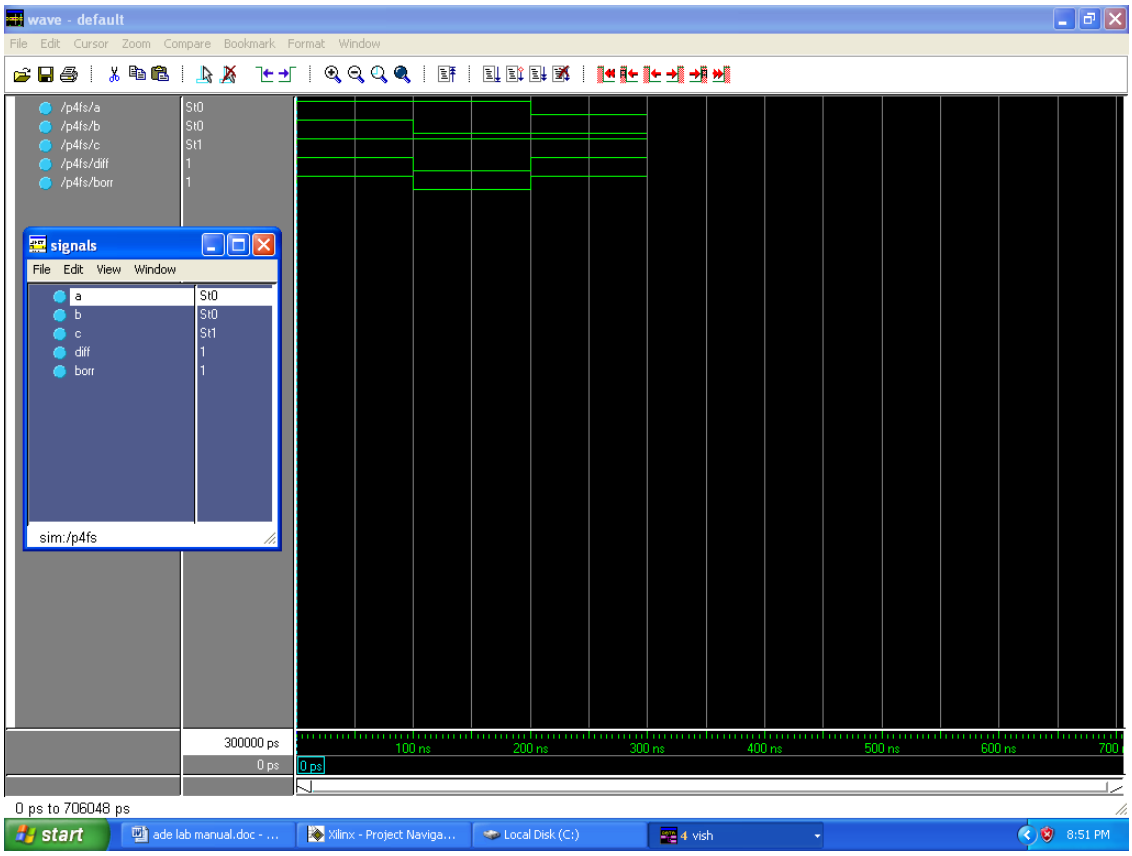
borr= $\bar{A}$ B + B C + $\bar{A}$ C

TRUTH TABLE

INPUTS			OUTPUTS	
A	B	C	diff	borr
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

```
module p4fs(a,b,c,diff,borr);
   input a;
   input b;
   input c;
   output diff;
   output borr;

reg diff, borr;
always @(a,b,c)
begin
diff = a^b^c;
borr= (~a & b) | (b & c) | (~a & c) ;
end
endmodule
```

OUTPUT: -

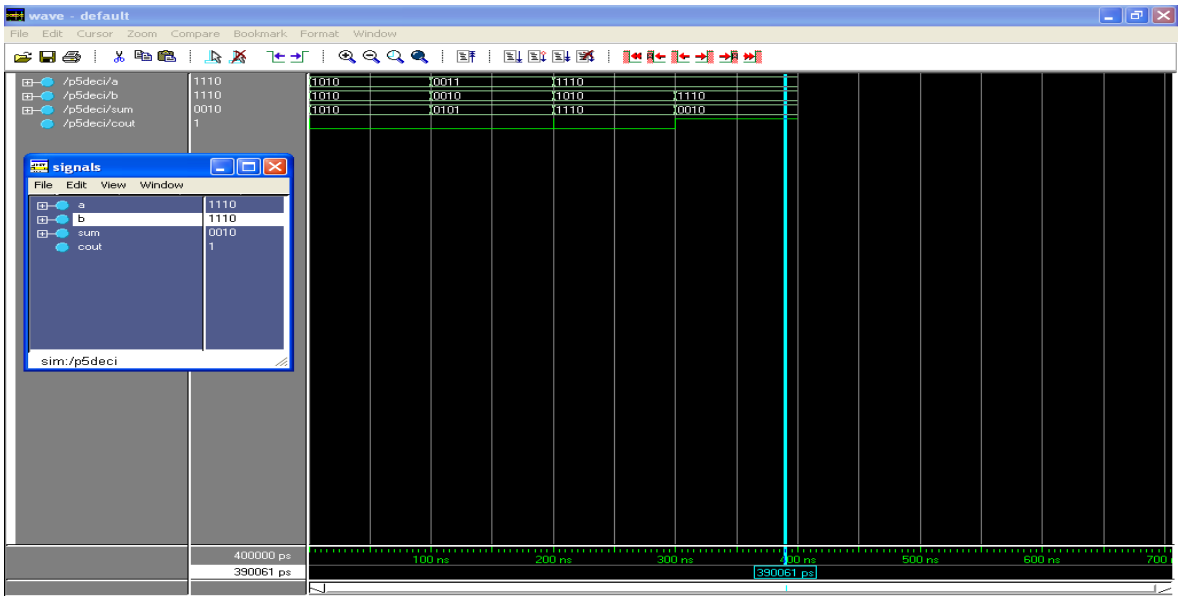

wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p4fs/a
/p4fs/b
/p4fs/c
/p4fs/diff
/p4fs/borr
St0
St0
St1
1
1
signals
File Edit View Window
a
b
c
diff
borr
St0
St0
St1
1
1
sim:/p4fs
300000 ps
0 ps
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
0 ps to 706048 ps
start
ade lab manual.doc - ...
Xilinx - Project Naviga...
Local Disk (C:)
4 vish
8:51 PM

5.Design Verilog HDL to implement Decimal adder.

This Verilog module, "DecimalAdder," takes two 4-bit decimal inputs A and B and produces a 4-bit sum (Sum) and a carry-out (CarryOut) output. The logic inside the "always" block performs decimal addition with carry propagation, and it also handles the case when the result is greater than 9. In such cases, it adds 6 to the result and updates the carry accordingly.

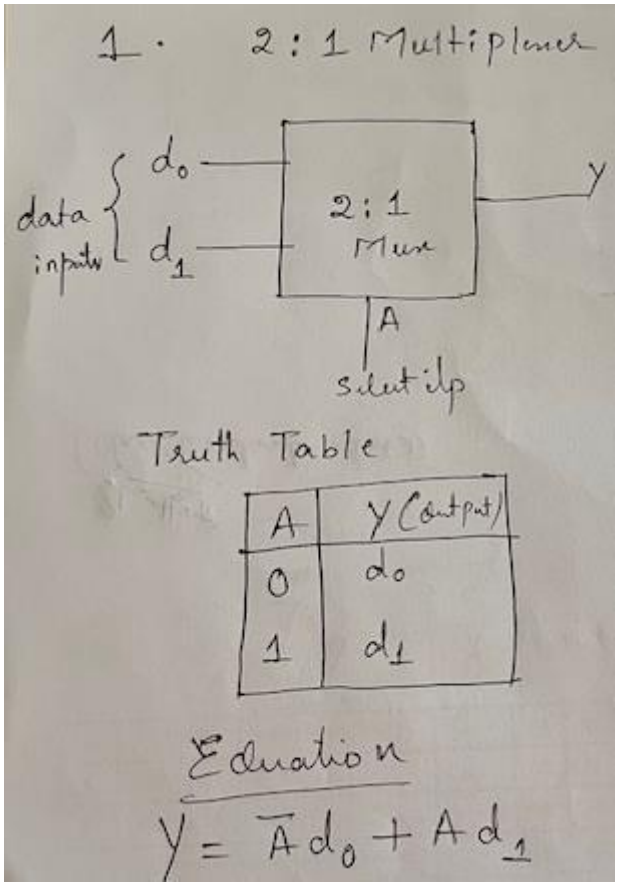
```
module p5deci(a,b,sum,cout);
  input [3:0] a;
  input [3:0] b;
  output [3:0] sum;
  output cout;

    reg [3:0] sum;
    reg cout;
    always@ (a,b)
    begin
    {cout,sum} = a+b;
    if(a>9 || b>9 || sum>9)
    begin
    {cout,sum} = sum+6;
    end
    end
    endmodule
```

OUTPUT: -



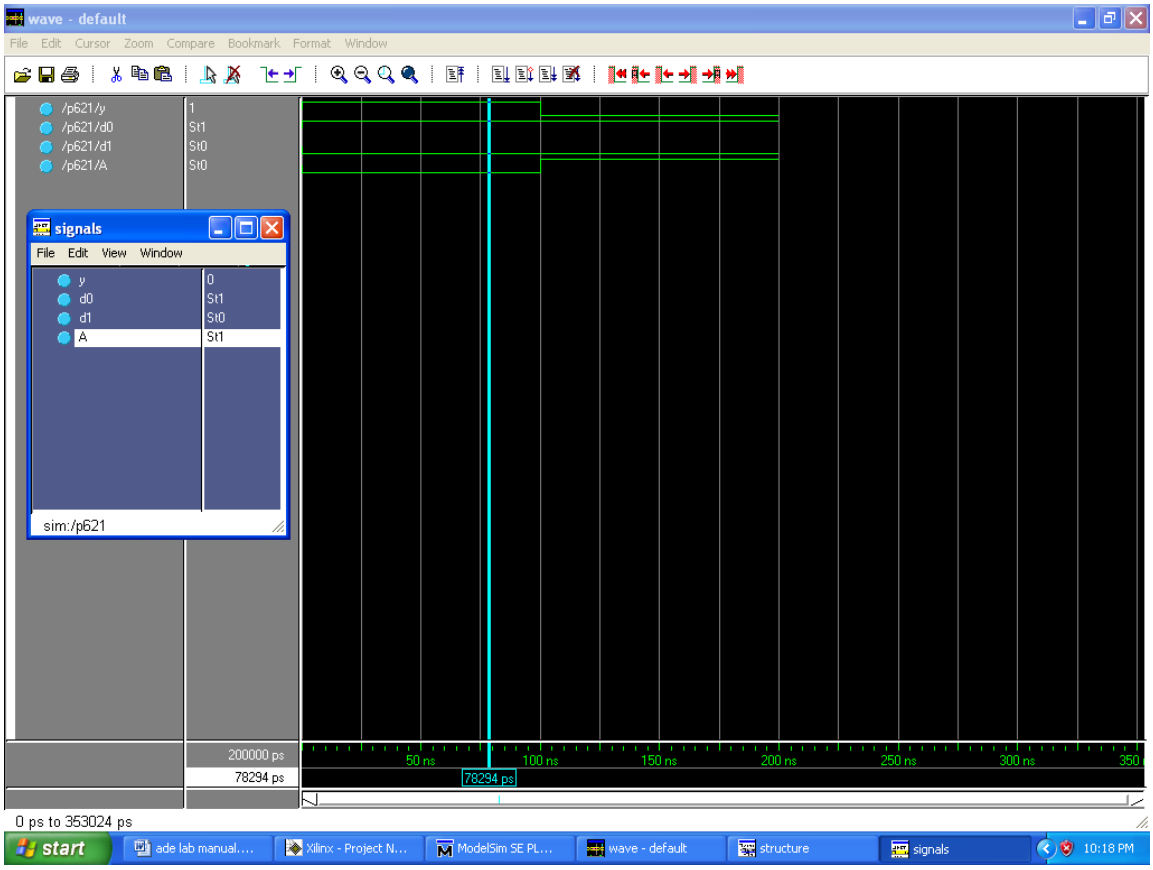
6.Design Verilog program to implement Different types of multiplexer like 2:1, 4:1 and 8:1.

2:1 Mux

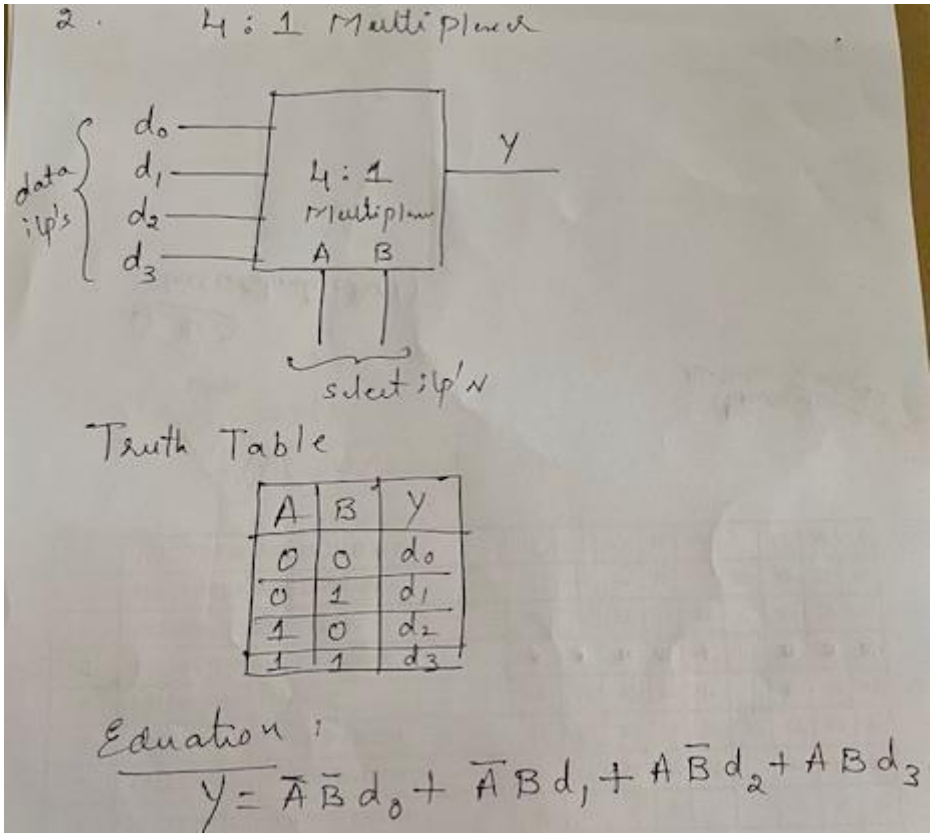



```
module p621(y,d0,d1,A);
   output y;
   input d0;
   input d1;
   input A;
   reg y;
   always @ (d0,d1,A)
   begin
   y=((~A & d0)|(A & d1));
   end
   endmodule
```

output: -

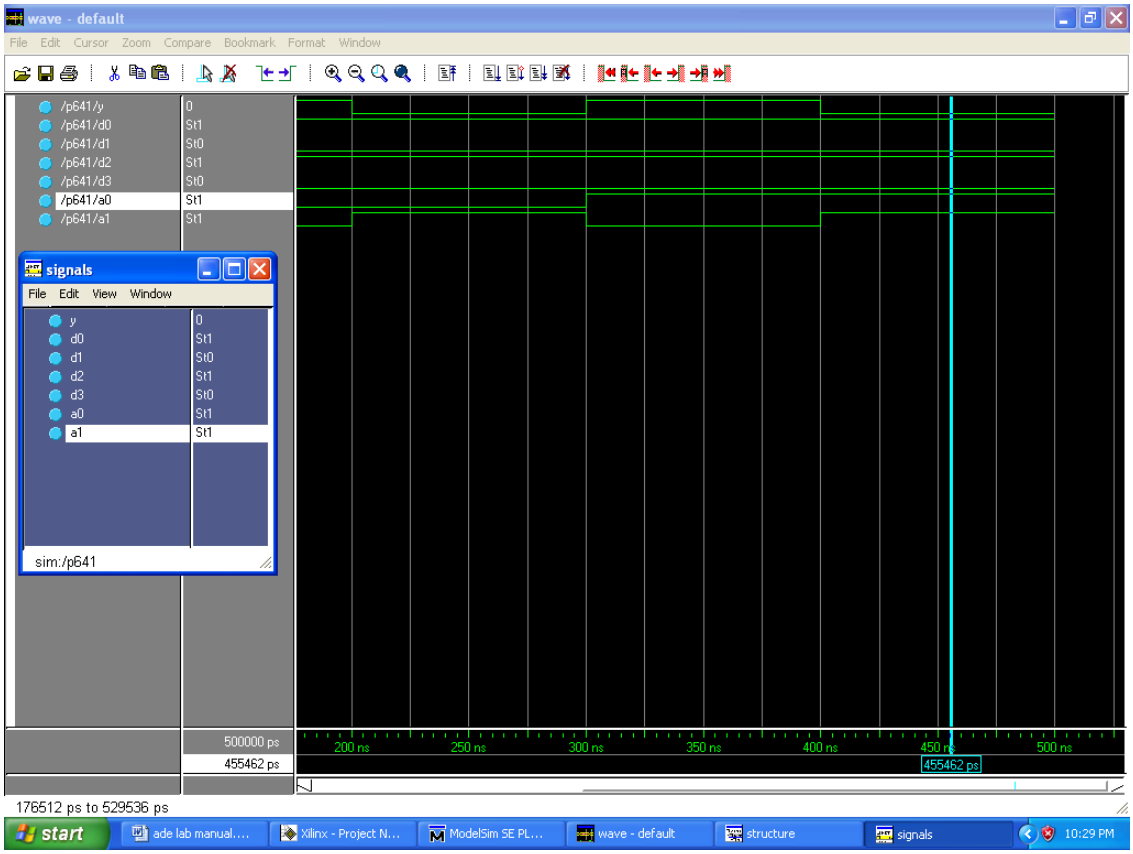

wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p621/y
/p621/d0
/p621/d1
/p621/A
1
St1
St0
St0
signals
File Edit View Window
y
d0
d1
A
0
St1
St0
St1
sim:/p621
200000 ps
78294 ps
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
78294 ps
0 ps to 353024 ps
start
ade lab manual....
Xilinx - Project N...
ModelSim SE PL...
wave - default
structure
signals
10:18 PM

4:1 Mux

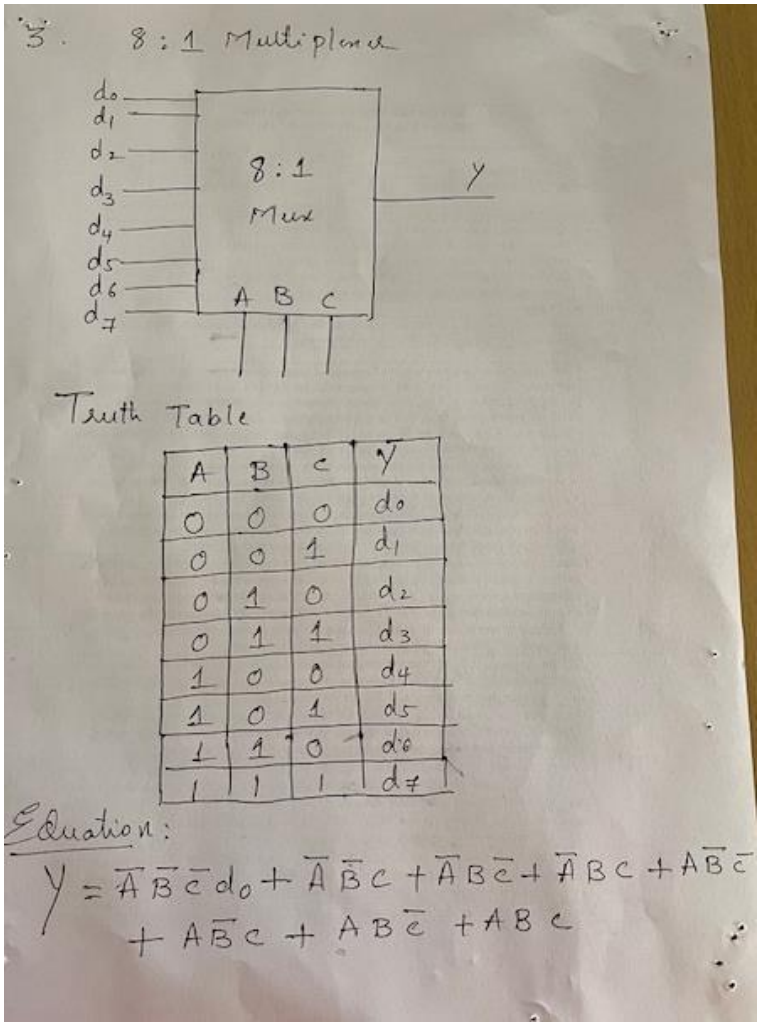



```
module p641(y,d0,d1,d2,d3,a0,a1);
  output y;
  input d0;
  input d1;
  input d2;
  input d3;
  input a0;
  input a1;

  reg y;
  always @ (d0,d1,d2,d3,a0,a1)
  begin
  y= (~a0 & ~a1 & d0) | (~a0 & a1 & d1) | (a0 & ~a1 & d2) | (a0 & a1 & d3);
  end
  endmodule
```

OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p641/y
/p641/d0
/p641/d1
/p641/d2
/p641/d3
/p641/a0
/p641/a1
signals
File Edit View Window
y
d0
d1
d2
d3
a0
a1
sim:/p641
500000 ps
455462 ps
200 ns
250 ns
300 ns
350 ns
400 ns
450 ns
500 ns
176512 ps to 529536 ps
start
ade lab manual....
Xilinx - Project N...
ModelSim SE PL...
wave - default
structure
signals
10:29 PM

8:1 Mux

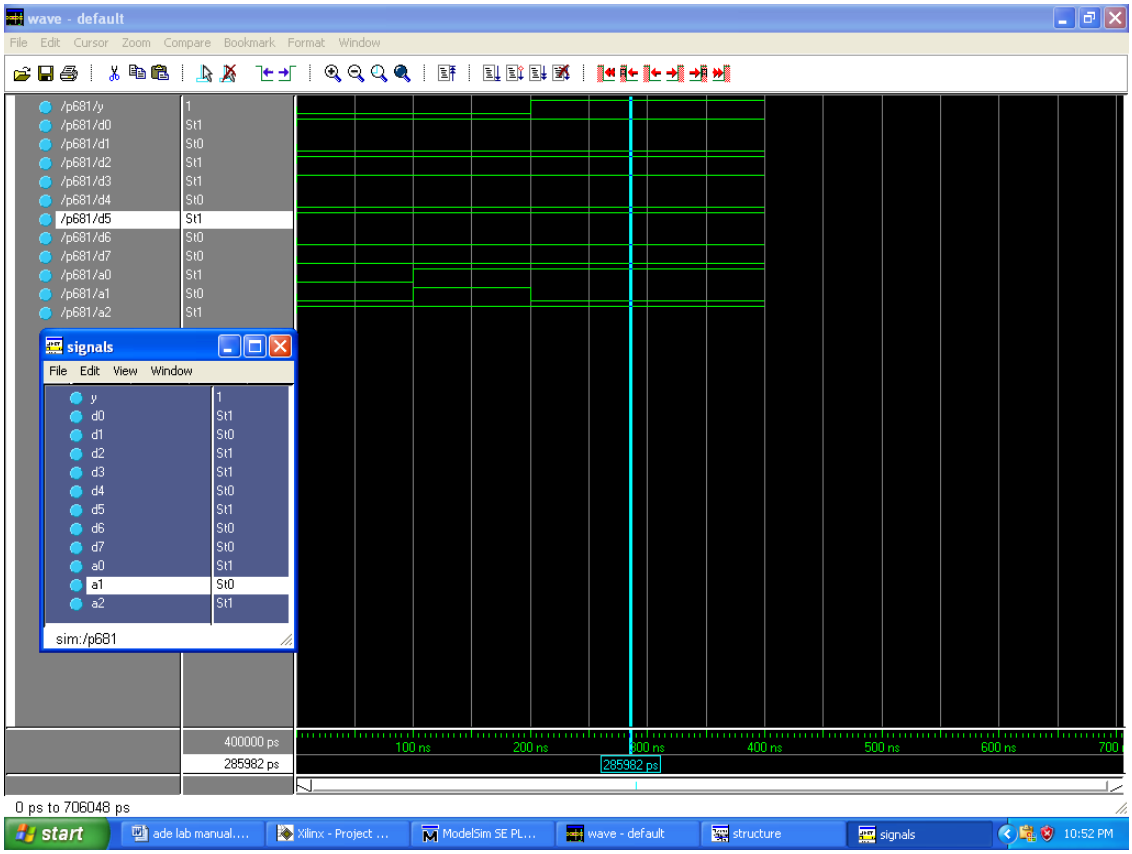


A	B	C	Y
0	0	0	d_0
0	0	1	d_1
0	1	0	d_2
0	1	1	d_3
1	0	0	d_4
1	0	1	d_5
1	1	0	d_6
1	1	1	d_7

```
module p681(y,d0,d1,d2,d3,d4,d5,d6,d7,a0,a1,a2);
   output y;
   input d0;
   input d1;
   input d2;
   input d3;
   input d4;
   input d5;
   input d6;
   input d7;
   input a0;
   input a1;
   input a2;
```

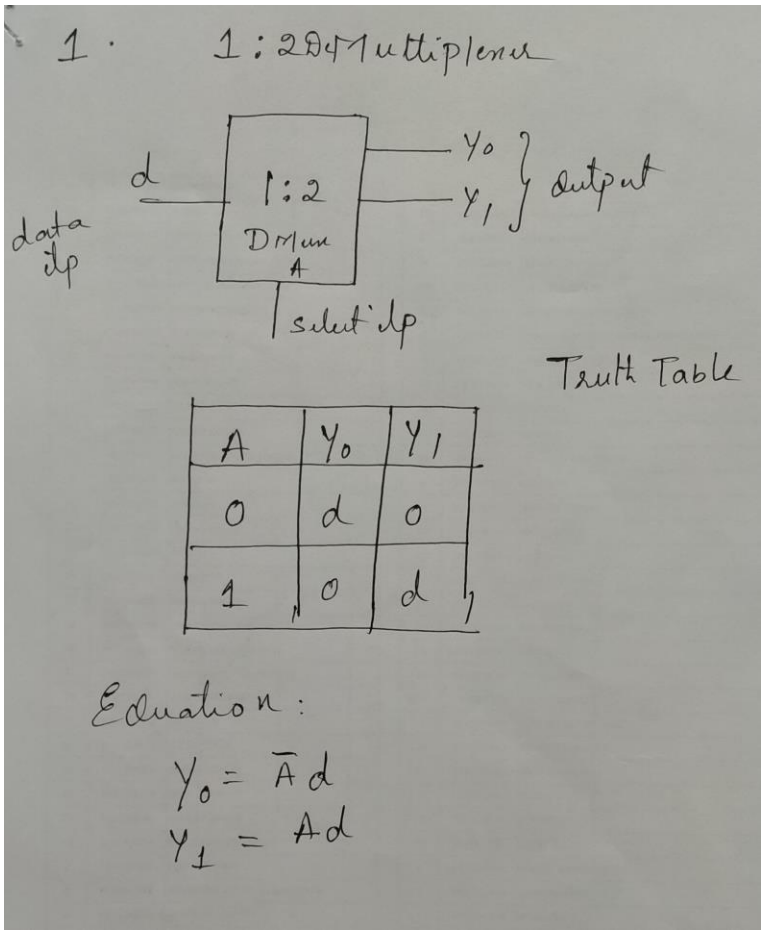
```
reg y;
always @ (d0,d1,d2,d3,d4,d5,d6,d7,a0,a1,a2)
begin

y= (~a0 & ~a1 & ~a2 & d0 ) |
   (~a0 & ~a1 & a2 & d1) |
   (~a0 & a1 & a2 & d2)  |
   ( ~a0 & a1 & a2 & d3)  |
   (a0 & ~a1 & ~a2 & d4) |
   (a0 & ~a1 & a2 & d5)  |
   (a0 & a1 & ~a2 & d6)  |
   (a0 & a1 & a2 & d7) ;
   end
   endmodule
```

OUTPUT: -



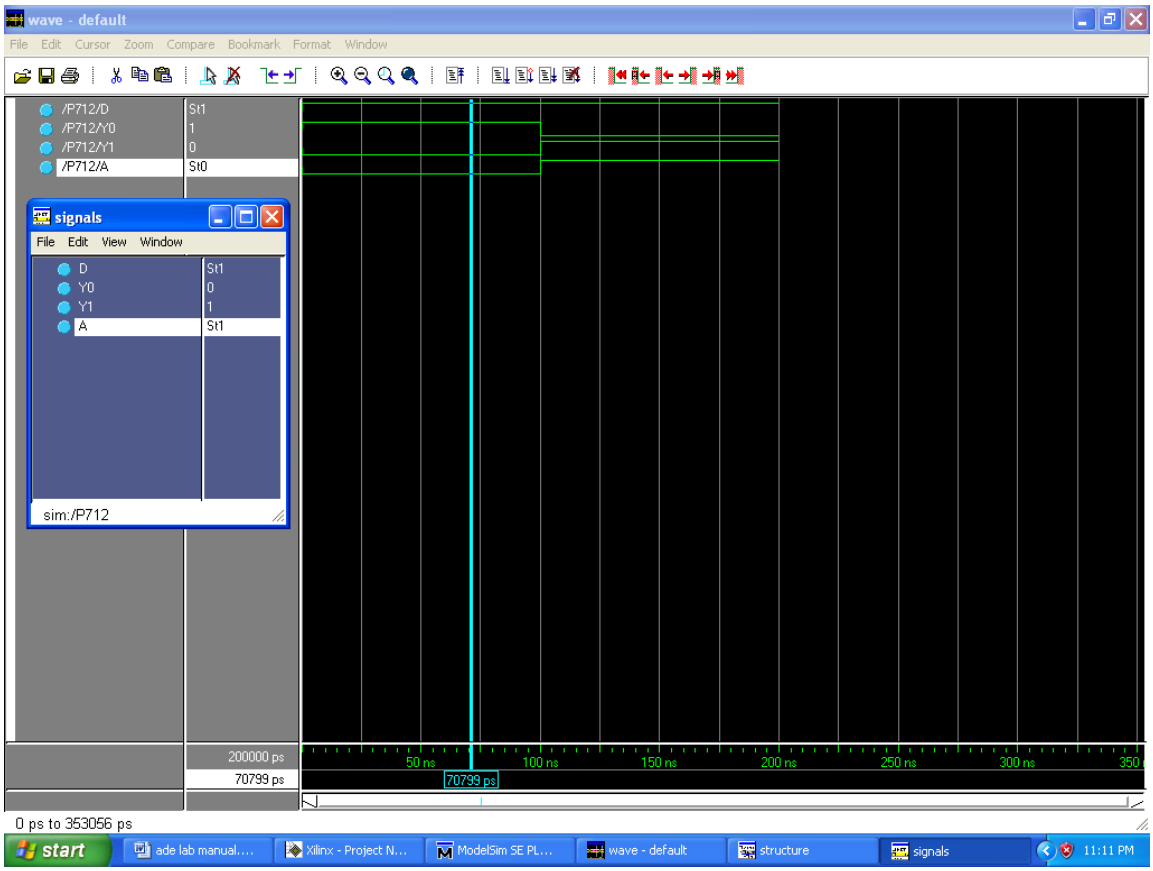
7.Design Verilog program to implement types of De-Multiplexer.

1:2 Demultiplexer

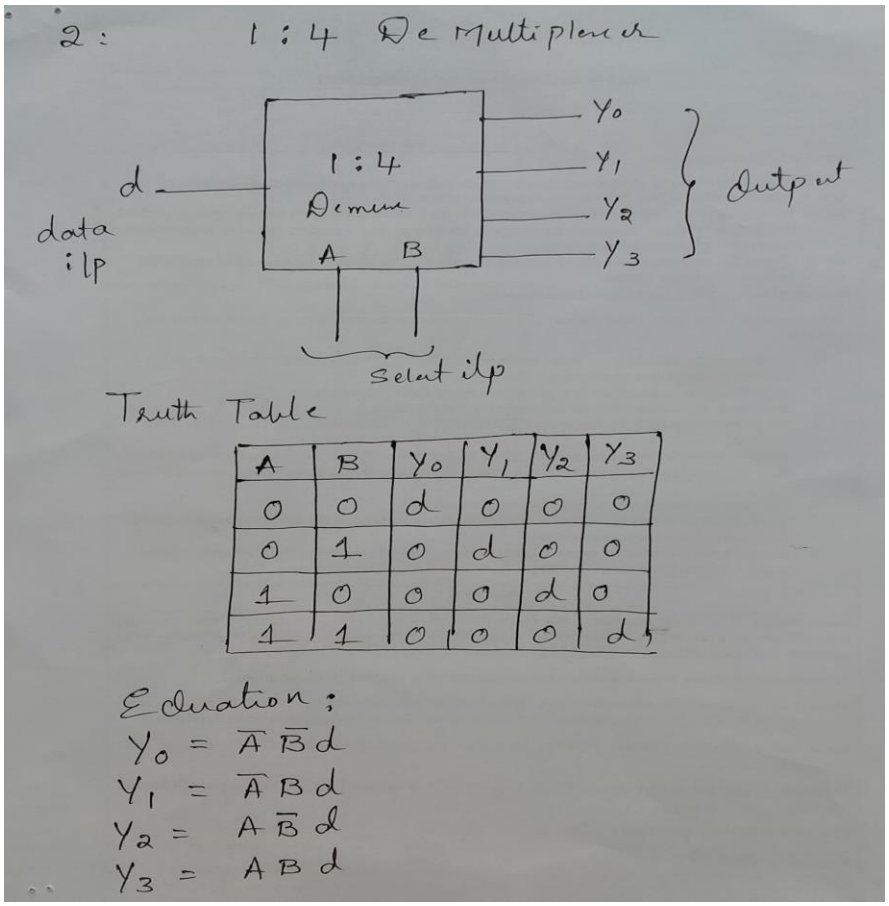



```
module P712(D,Y0,Y1,A);
input D;
output Y0;
output Y1;
input A;

    reg Y0,Y1;
    always @ (A,D)
    begin
    Y0=(~A & D);
    Y1=(A & D);
    end
    endmodule
```

OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/P712/D
/P712/Y0
/P712/Y1
/P712/A
St1
1
0
St0
signals
File Edit View Window
D
Y0
Y1
A
St1
0
1
St1
sim:/P712
200000 ps
70799 ps
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
70799 ps
0 ps to 353056 ps
start
ade lab manual....
Xilinx - Project N...
ModelSim SE PL...
wave - default
structure
signals
11:11 PM

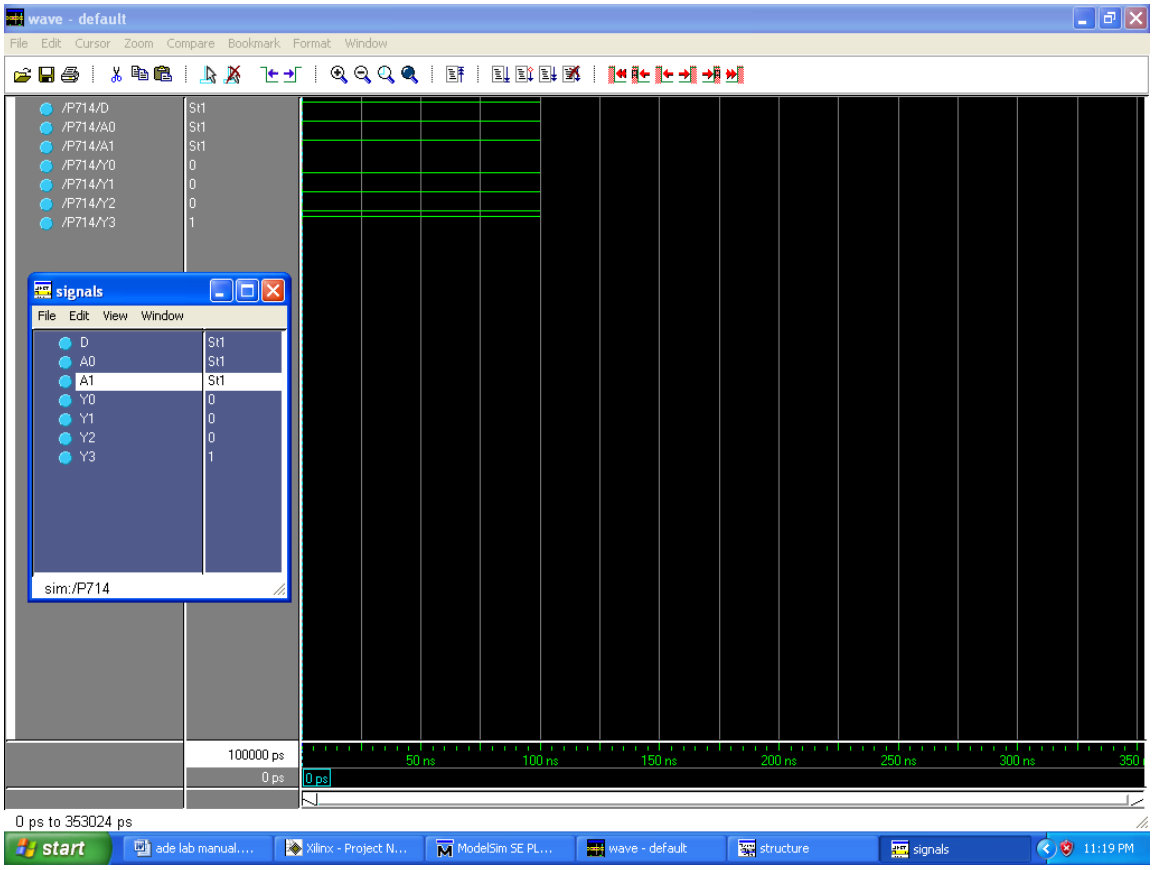
1:4 Demultiplexer



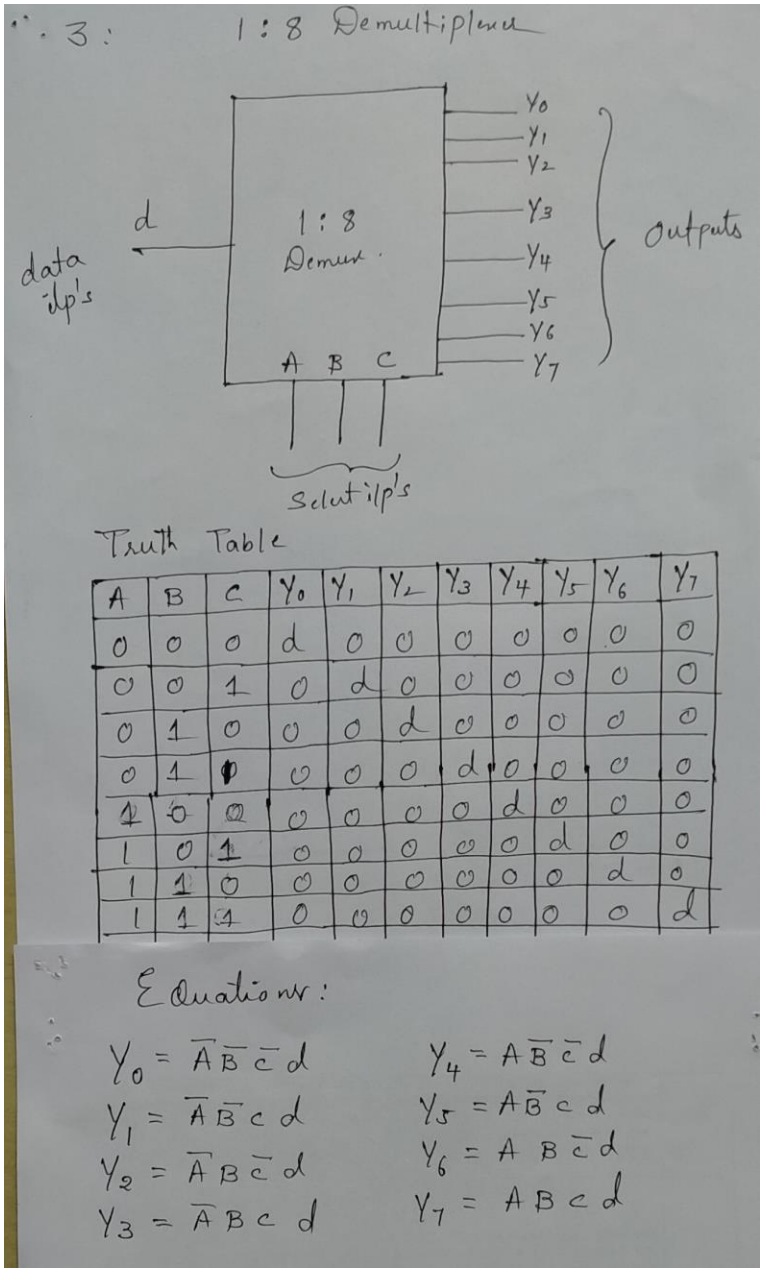
A	B	Y0	Y1	Y2	Y3
0	0	d	0	0	0
0	1	0	d	0	0
1	0	0	0	d	0
1	1	0	0	0	d

```
module P714(D,A0,A1,Y0,Y1,Y2,Y3);
  input D;
  input A0;
  input A1;
  output Y0;
  output Y1;
  output Y2;
  output Y3;

      reg Y0,Y1,Y2,Y3;
      always @ (A0,A1,D)
      begin
      Y0=(~A0 & ~A1 & D);
      Y1=(~A0 & A1 & D);
      Y2=(A0 & ~A1 & D);
      Y3=(A0 & A1 & D);
      end
      endmodule
```

OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/P714/D St1
/P714/A0 St1
/P714/A1 St1
/P714/Y0 0
/P714/Y1 0
/P714/Y2 0
/P714/Y3 1
signals
File Edit View Window
D St1
A0 St1
A1 St1
Y0 0
Y1 0
Y2 0
Y3 1
sim:/P714
100000 ps
0 ps
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
0 ps to 353024 ps
start
ade lab manual....
Xilinx - Project N...
ModelSim SE PL...
wave - default
structure
signals
11:19 PM

1:8 Demultiplexer



A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	d	0	0	0	0	0	0	0
0	0	1	0	d	0	0	0	0	0	0
0	1	0	0	0	d	0	0	0	0	0
0	1	1	0	0	0	d	0	0	0	0
1	0	0	0	0	0	0	d	0	0	0
1	0	1	0	0	0	0	0	d	0	0
1	1	0	0	0	0	0	0	0	d	0
1	1	1	0	0	0	0	0	0	0	d

```
module P718(D,A0,A1,A2,Y0,Y1,Y2,Y3,Y4,Y5,Y6,Y7);
   input D;
   input A0;
   input A1;
   input A2;
   output Y0;
   output Y1;
   output Y2;
```

output Y3;

output Y4;

output Y5;

output Y6;

output Y7;

reg Y0,Y1,Y2,Y3,Y4,Y5,Y6,Y7;

always @ (A0,A1,A2,D)

begin

Y0=(~A0 & ~A1 & ~A2 & D);

Y1=(~A0 & ~A1 & A2 & D);

Y2=(~A0 & A1 & ~A2 & D);

Y3=(~A0 & A1 & A2 & D);

Y4=(A0 & ~A1 & ~A2 & D);

Y5=(A0 & ~A1 & A2 & D);

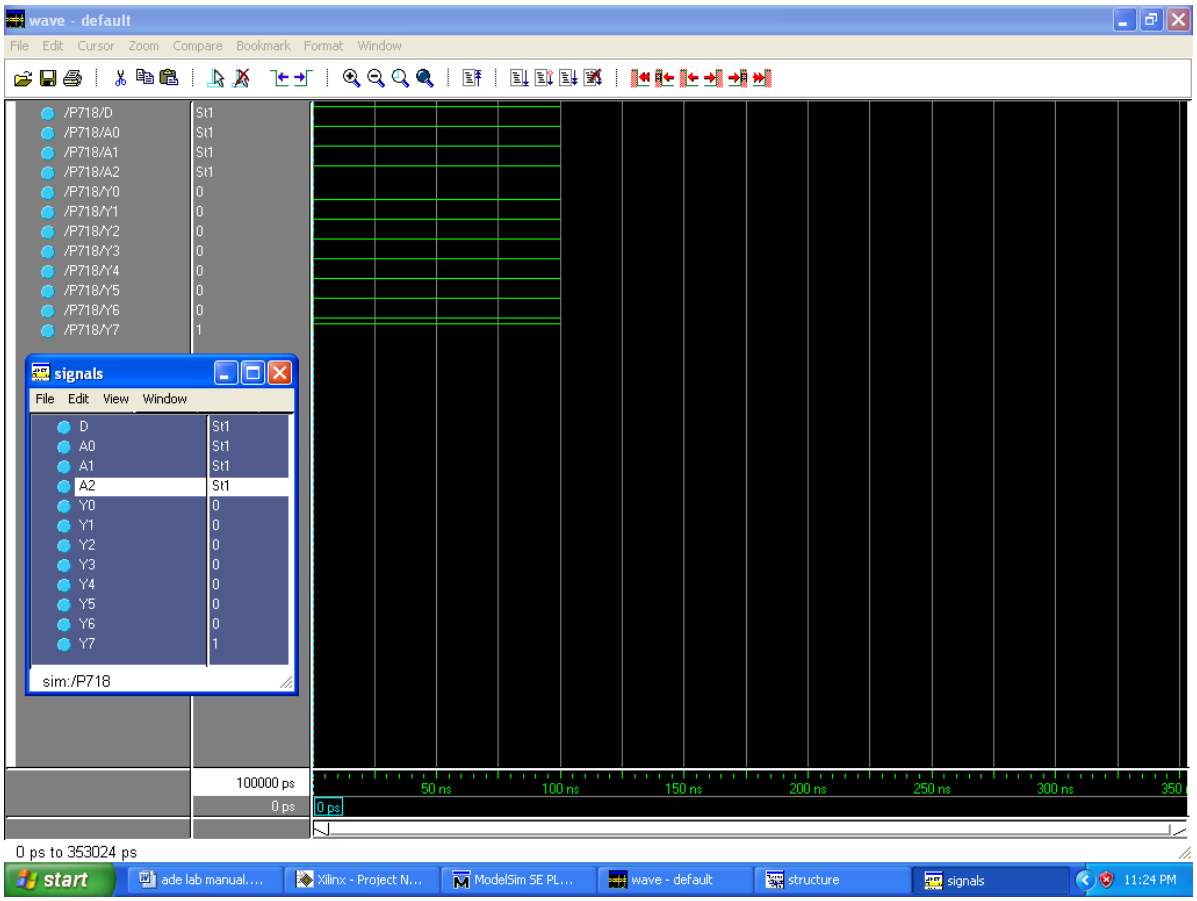
Y6=(A0 & A1 & ~A2 & D);

Y7=(A0 & A1 & A2 & D);

end

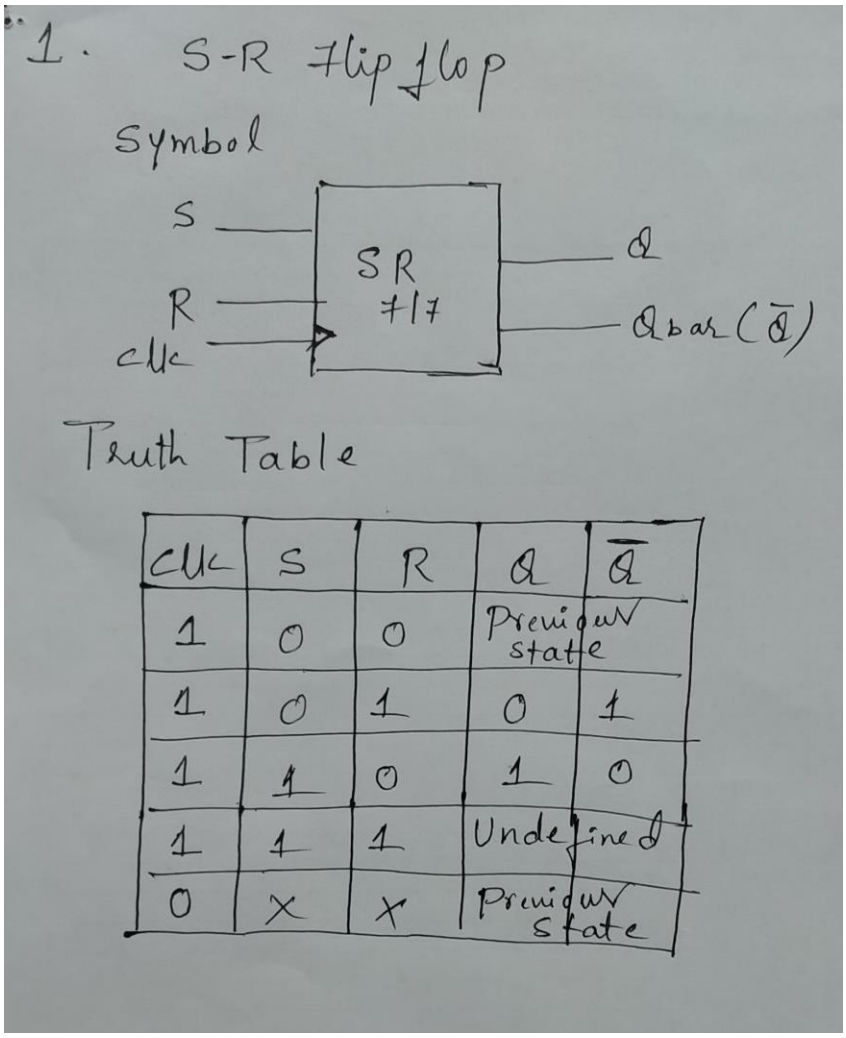
endmodule

OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/P718/D St1
/P718/A0 St1
/P718/A1 St1
/P718/A2 St1
/P718/Y0 0
/P718/Y1 0
/P718/Y2 0
/P718/Y3 0
/P718/Y4 0
/P718/Y5 0
/P718/Y6 0
/P718/Y7 1
signals
File Edit View Window
D St1
A0 St1
A1 St1
A2 St1
Y0 0
Y1 0
Y2 0
Y3 0
Y4 0
Y5 0
Y6 0
Y7 1
sim:/P718
100000 ps
0 ps
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
0 ps to 353024 ps
start
ade lab manual....
Xilinx - Project N...
ModelSim SE PL...
wave - default
structure
signals
11:24 PM

8.Design Verilog program for implementing various types of Flip-Flops such as SR, JK and D.

SR FLIP-FLOP

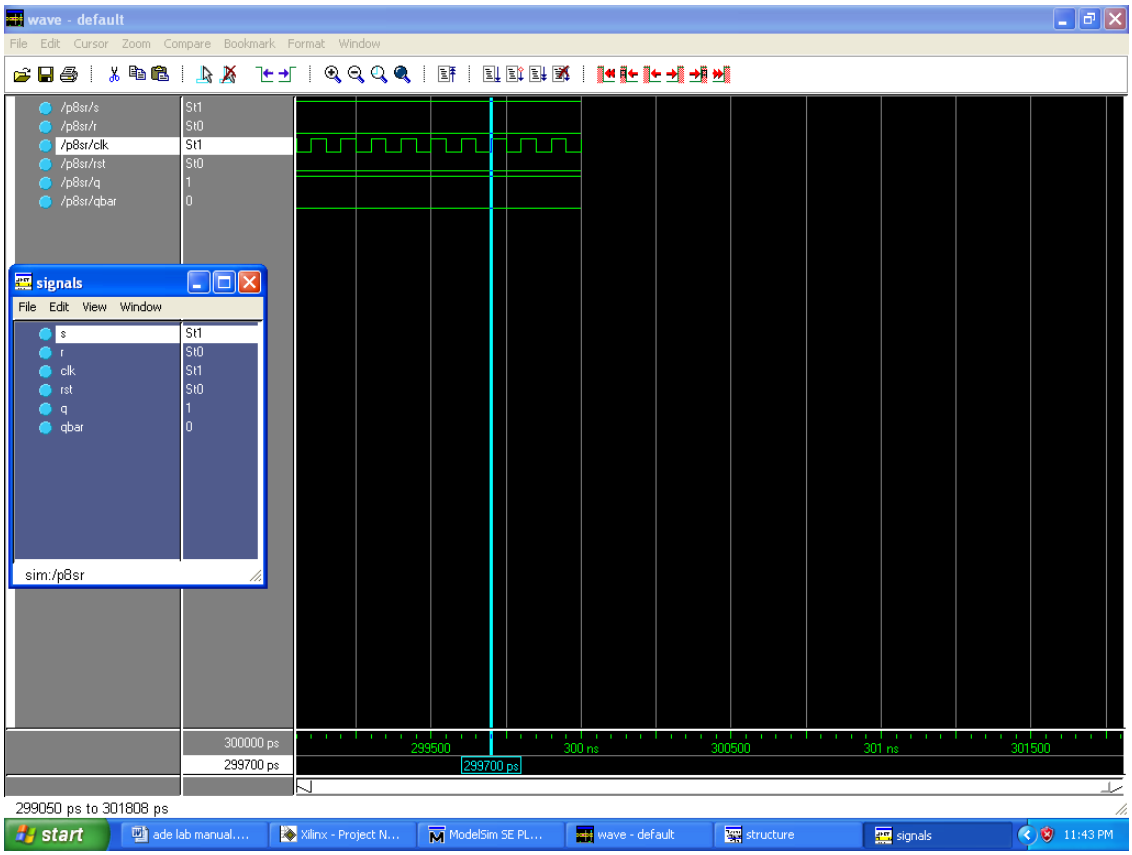


clk	S	R	Q	$\bar{Q}$
1	0	0	Previous state	
1	0	1	0	1
1	1	0	1	0
1	1	1	Undefined	
0	X	X	Previous state	

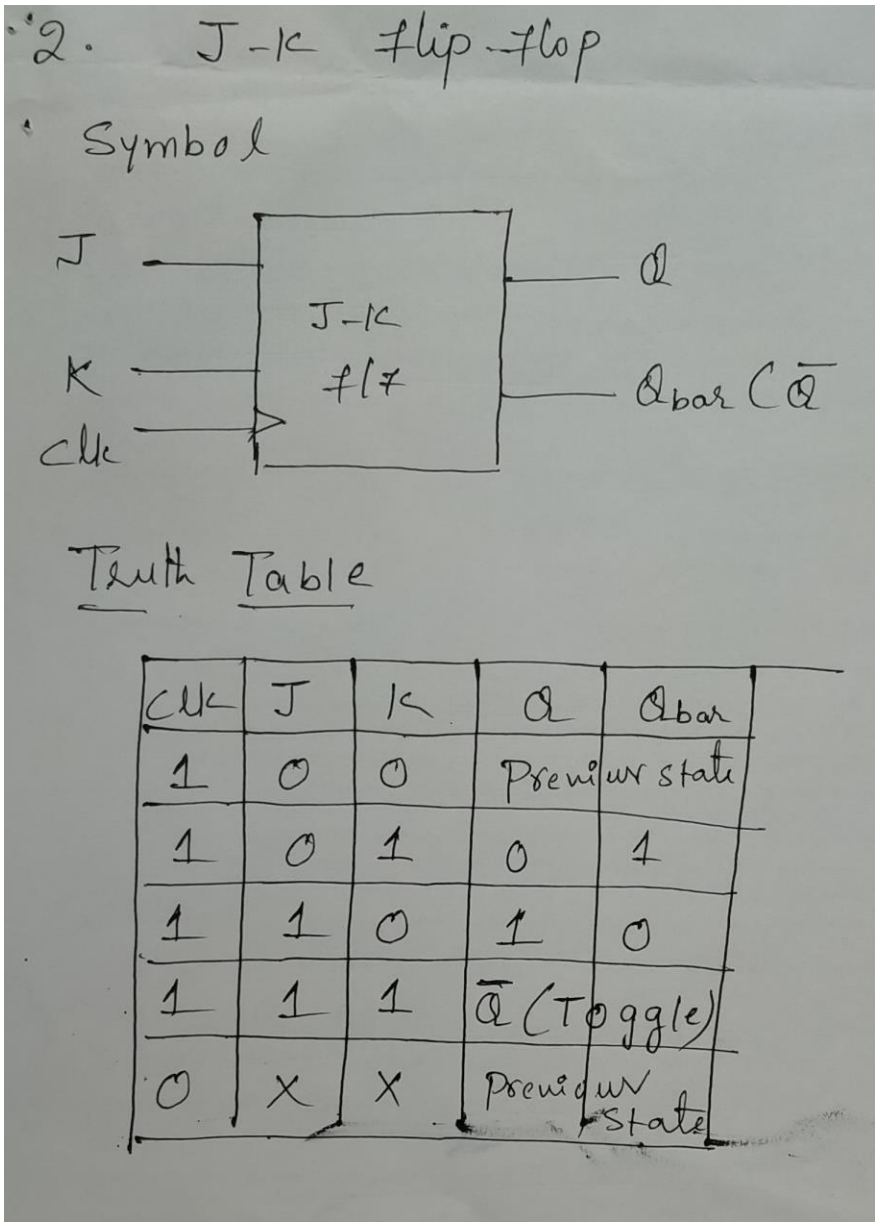
```
module p8sr(s,r,clk,rst,q,qbar);
   input s;
   input r;
   input clk;
   input rst;
   output q;
   output qbar;
```

```
 reg q,qbar;
 always@(posedge clk)
 begin
 if(rst)
 q<=1'b0;
else if (s==1'b0 && r==1'b0) q<=q;
else if (s==1'b0 && r==1'b1) q<=1'b0;
else if (s==1'b1 && r==1'b0) q<=1'b1;
else if (s==1'b1 && r==1'b1) q<=1'bx;
assign qbar=~q;
end
endmodule
```

OUTPUT: -



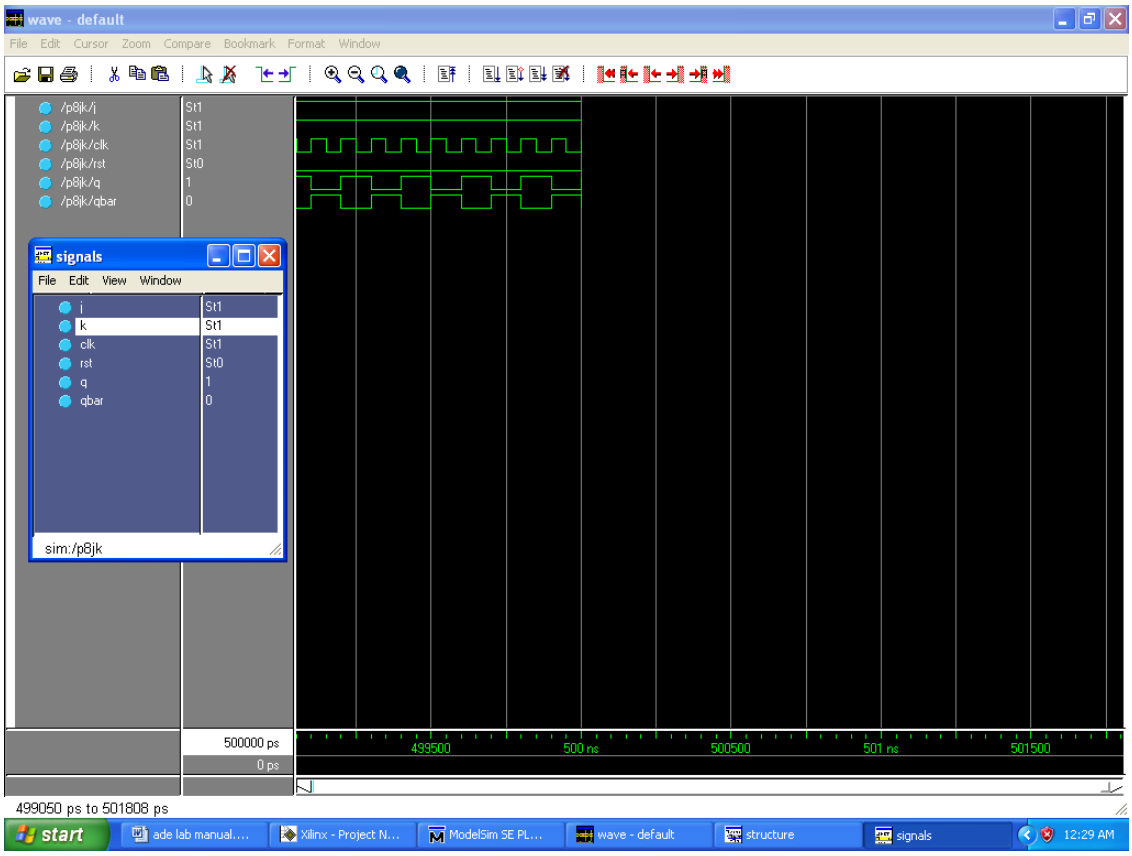
JK FLIP-FLOP

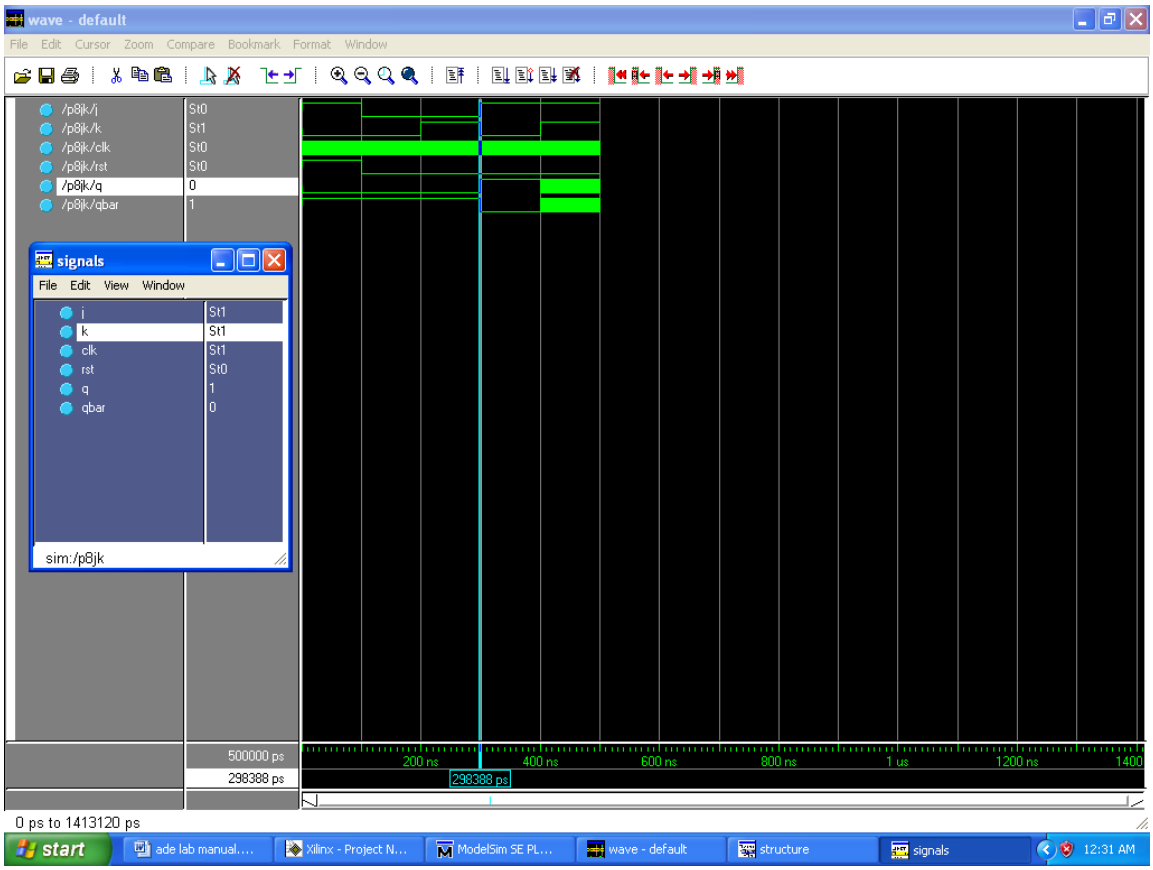



```
module p8jk(j,k,clk,rst,q,qbar);
   input j;
   input k;
   input clk;
   input rst;
   output q;
   output qbar;
```

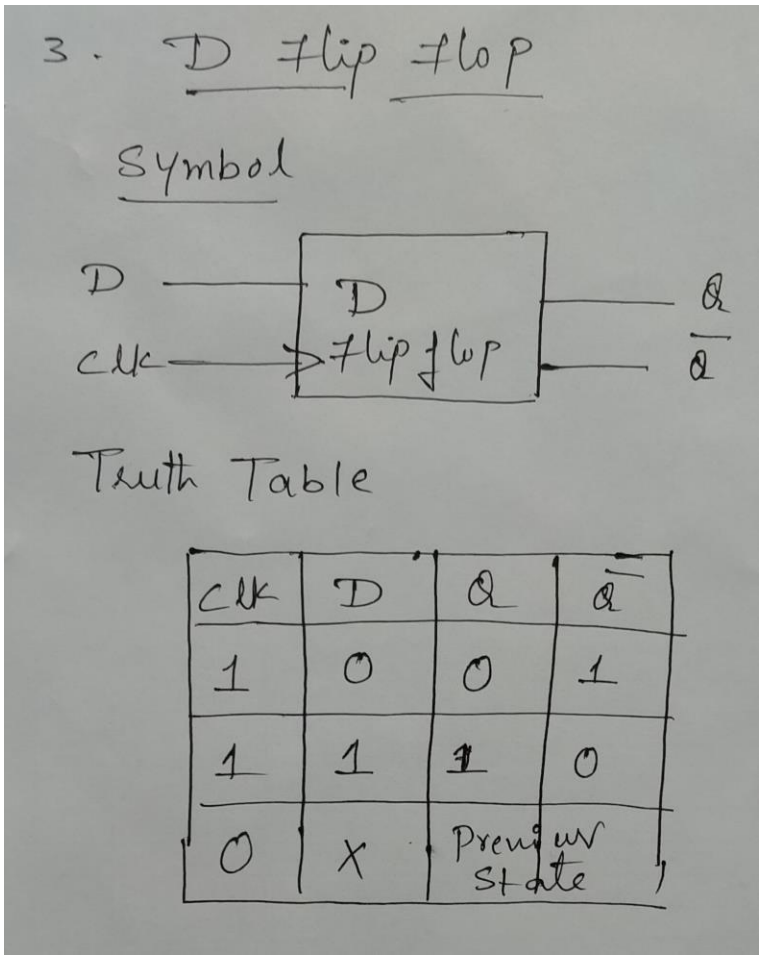
```
 reg q,qbar;
 always@(posedge clk)
 begin
 if(rst)
 q<=1'b0;
else if (j==1'b0 && k==1'b0) q<=q;
else if (j==1'b0 && k==1'b1) q<=1'b0;
else if (j==1'b1 && k==1'b0) q<=1'b1;
else if (j==1'b1 && k==1'b1) q<= ~q;
assign qbar = ~q;
end
endmodule
```

OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p8jk/j St1
/p8jk/k St1
/p8jk/clk St1
/p8jk/rst St0
/p8jk/q 1
/p8jk/qbar 0
signals
File Edit View Window
j St1
k St1
clk St1
rst St0
q 1
qbar 0
sim:/p8jk
500000 ps
0 ps
499500
500 ns
500500
501 ns
501500
499050 ps to 501808 ps
start
ade lab manual....
Xilinx - Project N...
ModelSim SE PL...
wave - default
structure
signals
12:29 AM

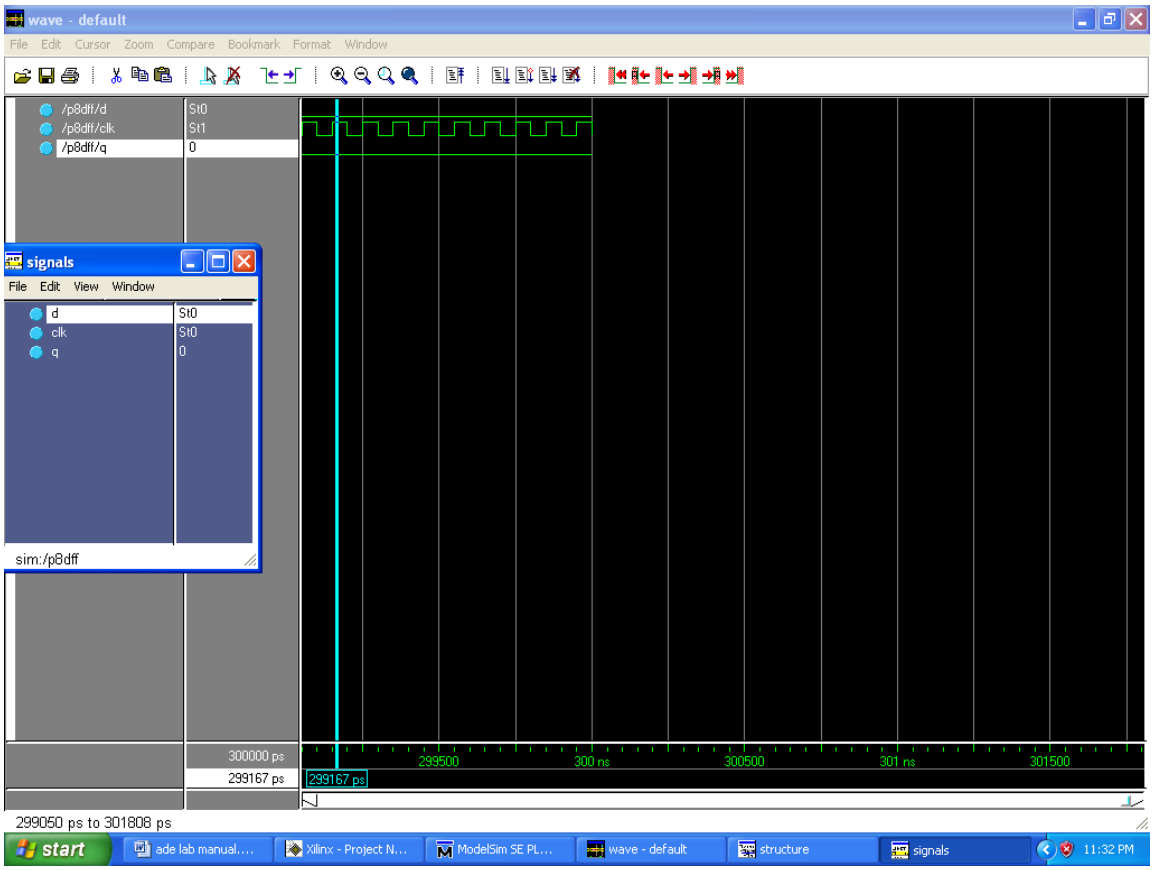

wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p8jk/j St0
/p8jk/k St1
/p8jk/clk St0
/p8jk/rst St0
/p8jk/q 0
/p8jk/qbar 1
signals
File Edit View Window
j St1
k St1
clk St1
rst St0
q 1
qbar 0
sim:/p8jk
500000 ps
298388 ps
200 ns
400 ns
600 ns
800 ns
1 us
1200 ns
1400
298388 ps
0 ps to 1413120 ps
start
ade lab manual....
Xilinx - Project N...
ModelSim SE PL...
wave - default
structure
signals
12:31 AM

D FLIP-FLOP




```
module p8dff(d,clk,q);
   input d;
   input clk;
   output q;

reg q;
always @(posedge clk)
begin
q<=d;
end
endmodule
```

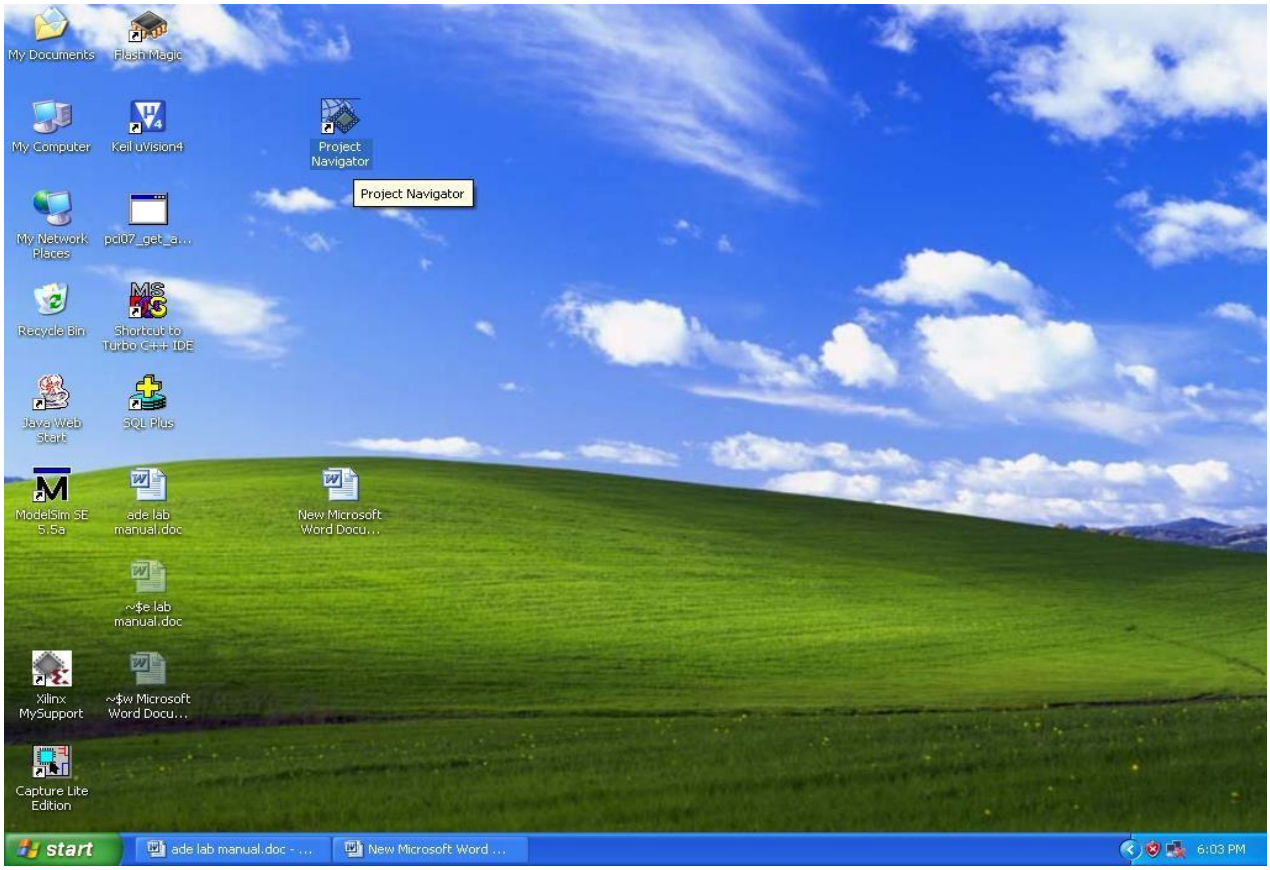
OUTPUT: -


wave - default
File Edit Cursor Zoom Compare Bookmark Format Window
/p8dff/d
/p8dff/clk
/p8dff/q
St0
St1
0
signals
File Edit View Window
d
clk
q
St0
St0
0
sim:/p8dff
300000 ps
299167 ps
299500
300 ns
300500
301 ns
301500
299167 ps
299050 ps to 301808 ps
start
ade lab manual....
Xilinx - Project N...
ModelSim SE PL...
wave - default
structure
signals
11:32 PM

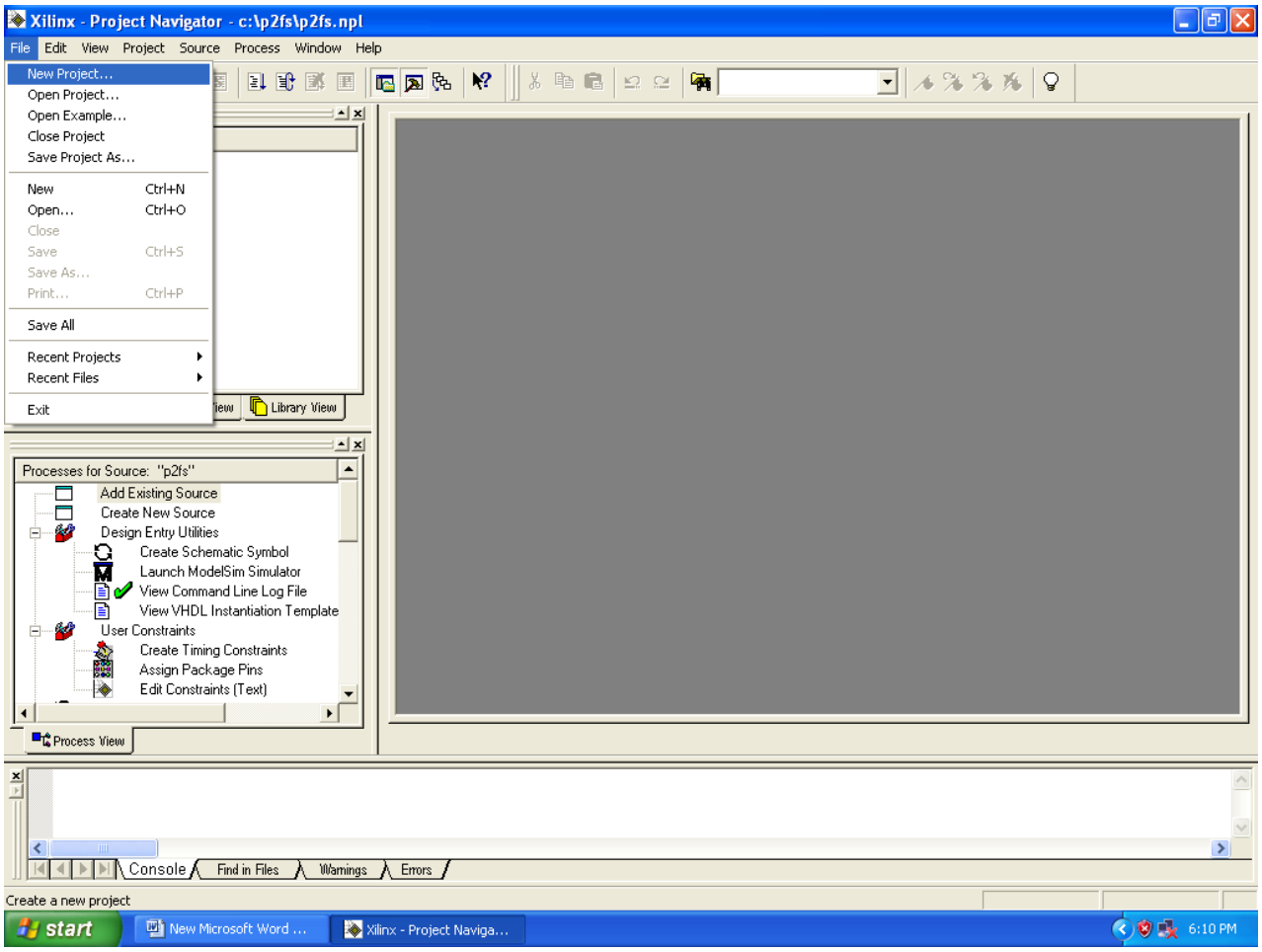
STEPS TO BE FOLLOWED FOR EXECUTING

THE VERILOG PROGRAMS USING XILINX SOFTWARE

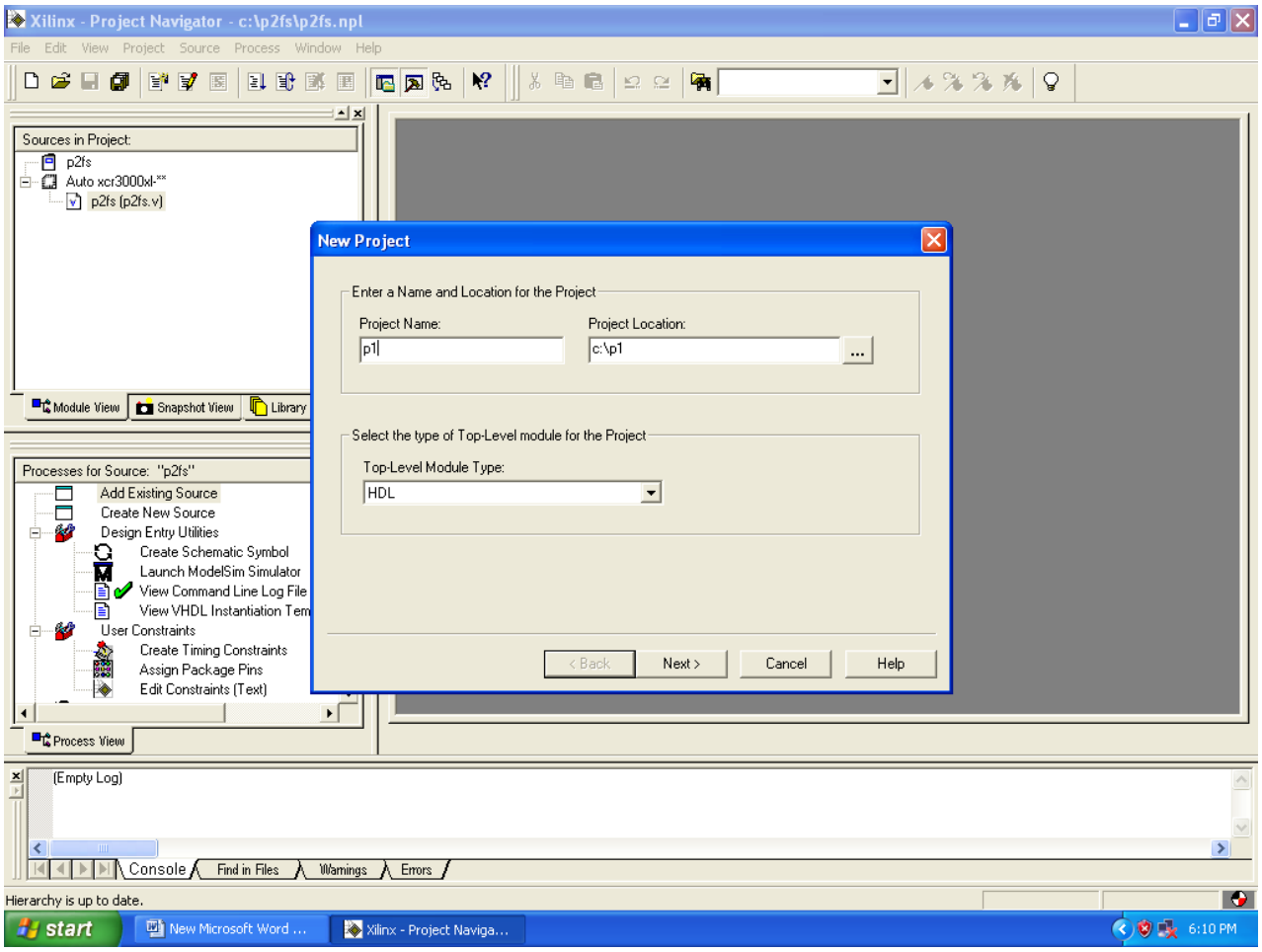
1. click on the project navigator.



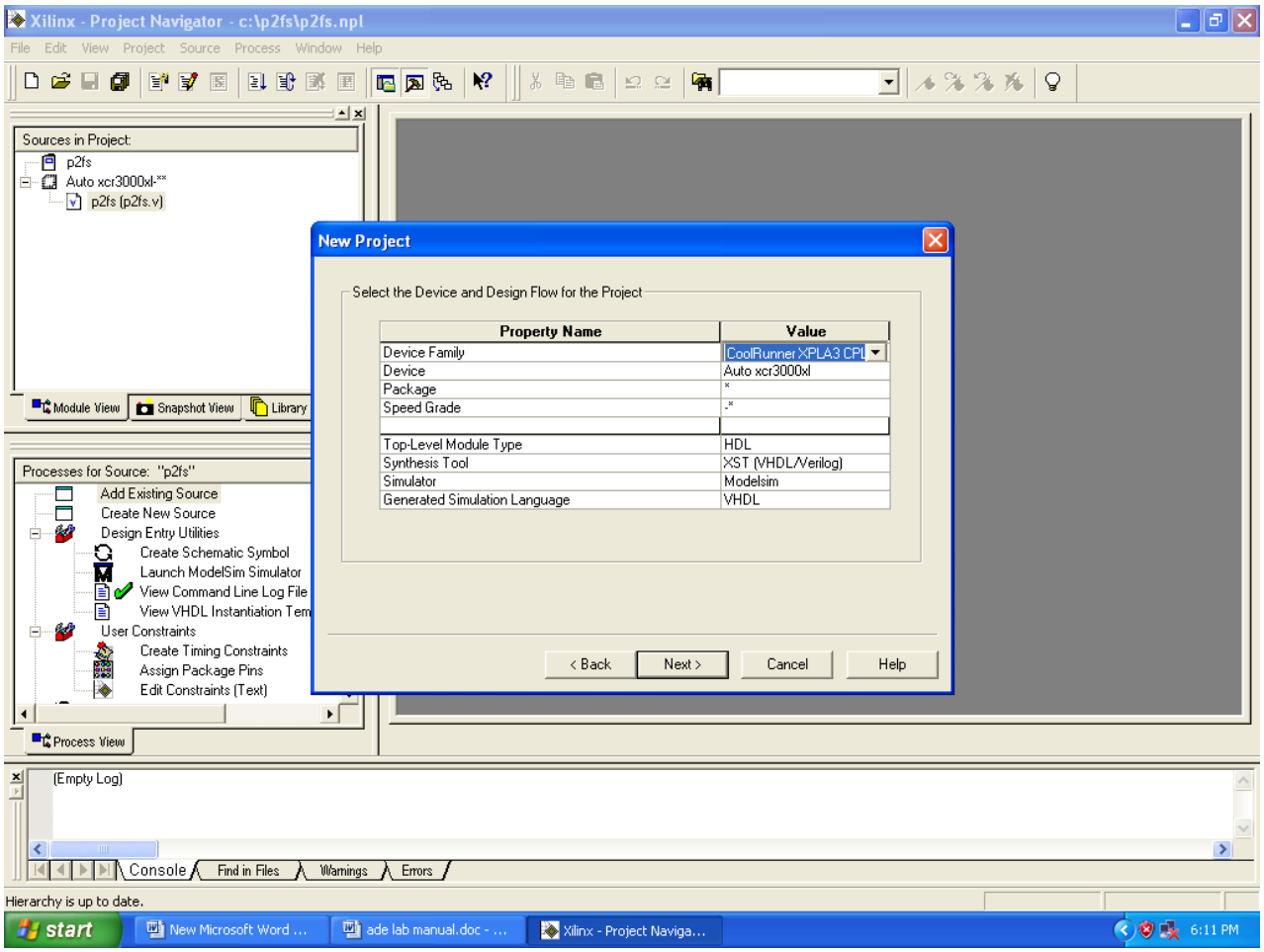
2. go to the file and select new project.



3. enter the project name and click next.



4. select device family cool runner XPLA3 CPL (by default) click next.



5. click on new source.