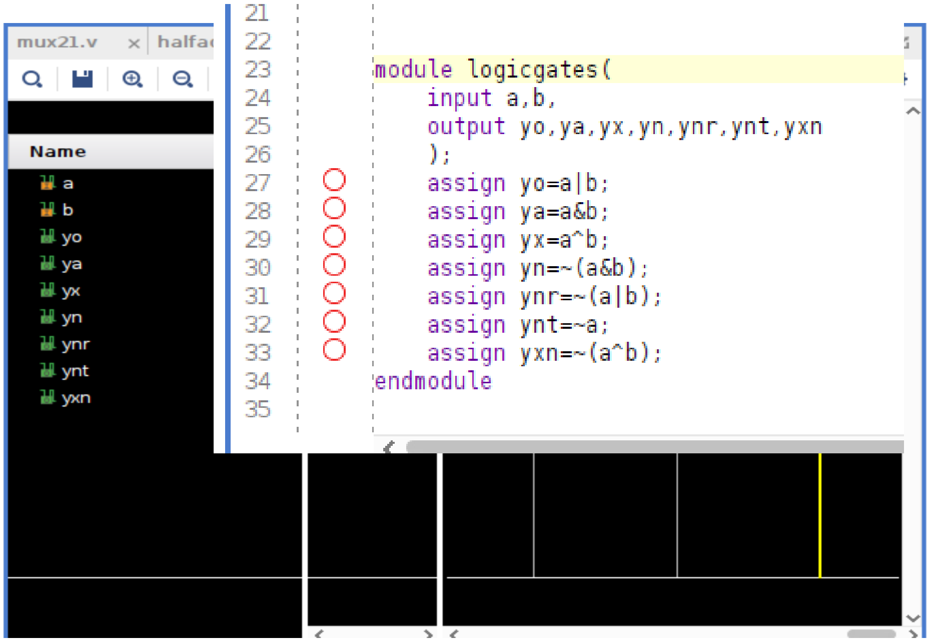
Digital system prototyping using FPGA

assignment 1

SHAMIL K

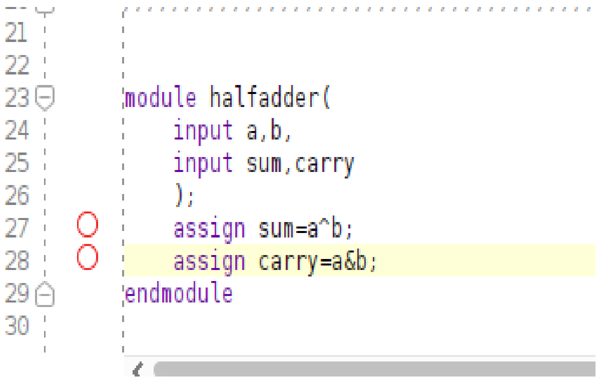
1. LOGIC GATES



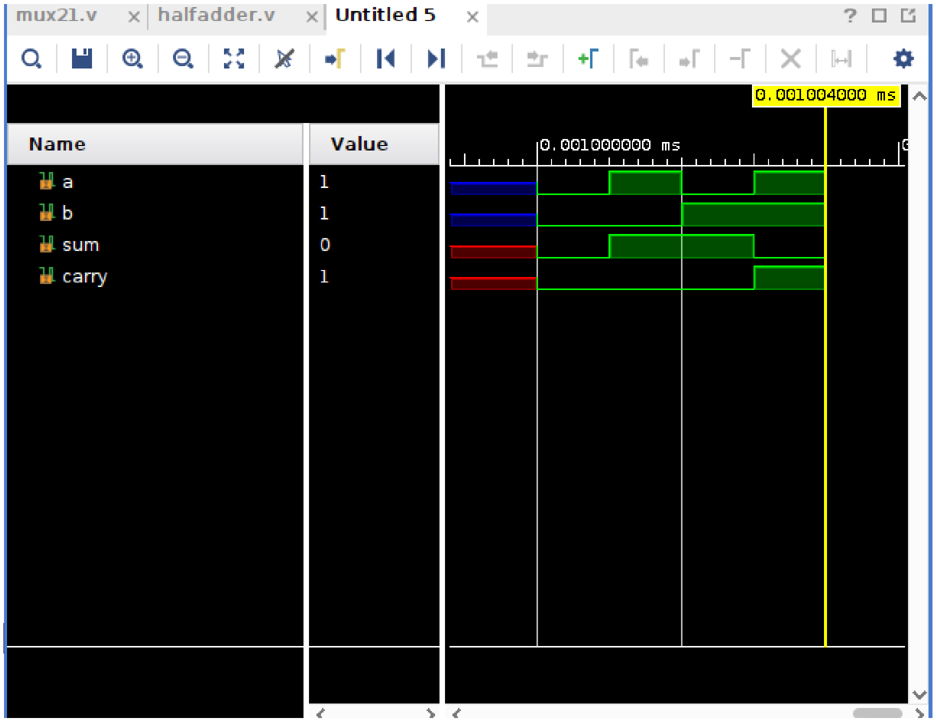
```
module logicgates(
    input a,b,
    output yo,ya,yx,yn,ynr,ynt,yxn
    );
    assign yo=a|b;
    assign ya=a&b;
    assign yx=a^b;
    assign yn=~(a&b);
    assign ynr=~(a|b);
    assign ynt=~a;
    assign yxn=~(a^b);
endmodule
```

2.ADDERS

a. HALF ADDER

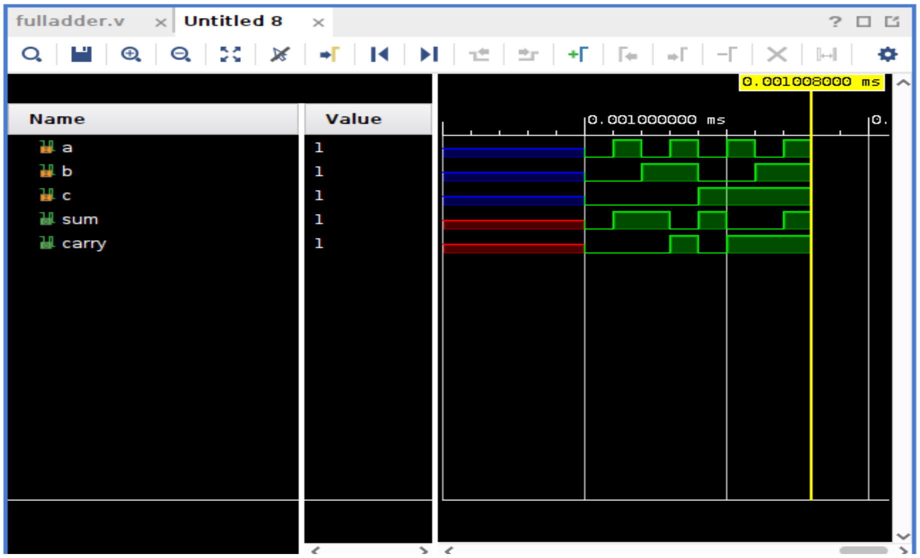


```
module halfadder(
    input a,b,
    input sum,carry
    );
    assign sum=a^b;
    assign carry=a&b;
endmodule
```



Name	Value
a	1
b	1
sum	0
carry	1

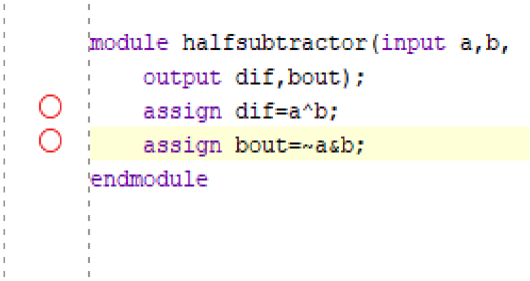
B.Ful



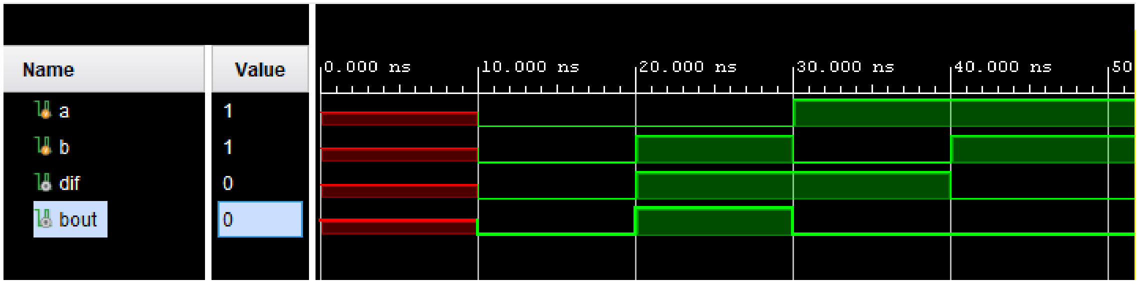
Name	Value
a	1
b	1
c	1
sum	1
carry	1

3.SUBTRACTORS

a. HALF SUBTRACTC

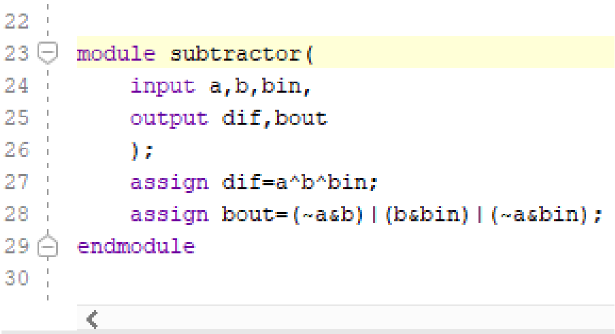


```
module halfsubtractor(input a,b,
    output dif,bout);
    assign dif=a^b;
    assign bout=~a&b;
endmodule
```

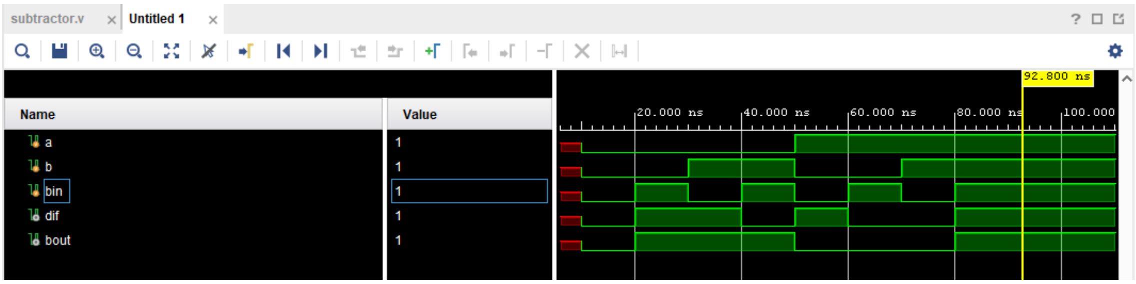


Name	Value
a	1
b	1
dif	0
bout	0

b. FULL SUBTRACTOR

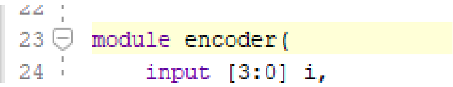


```
module subtractor(
    input a,b,bin,
    output dif,bout
    );
    assign dif=a^b^bin;
    assign bout=(~a&b)|(b&bin)|(~a&bin);
endmodule
```



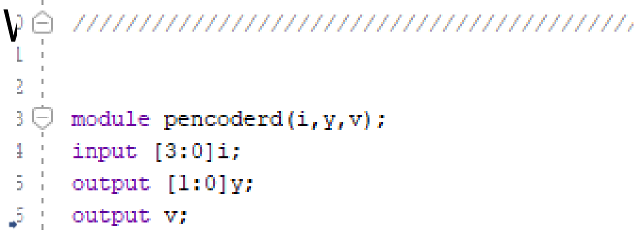
Name	Value
a	1
b	1
bin	1
dif	1
bout	1

5. PRIORITY ENC

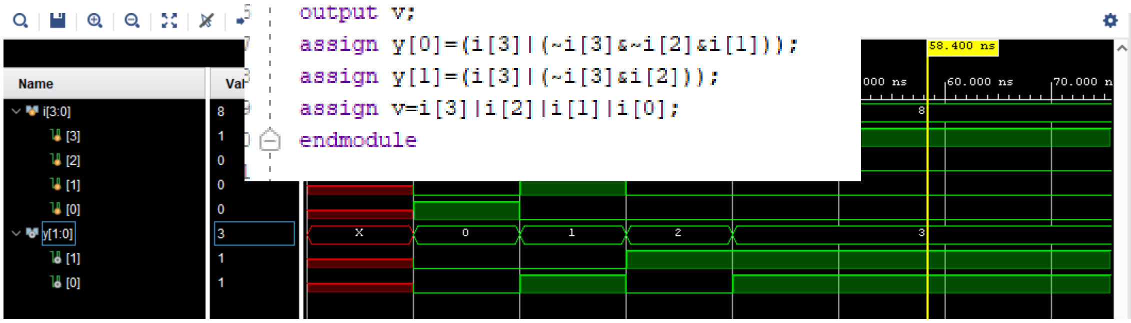


```
module encoder(
    input [3:0] i,
```

a. DATAFLOV

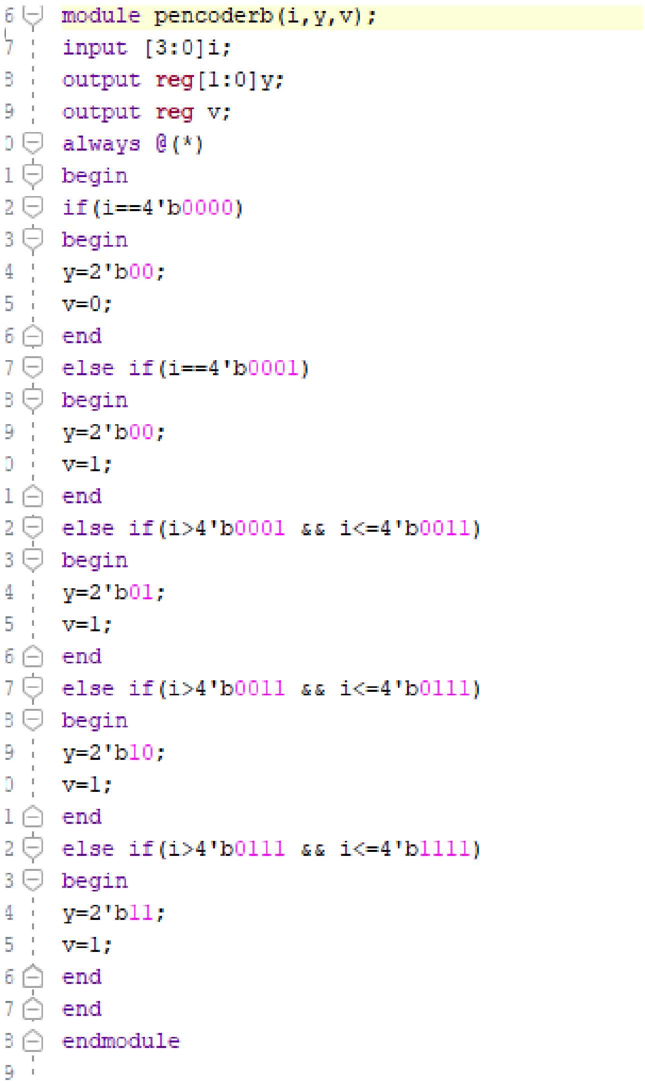


```
/////////////////////////////////////////////////////////////


module pencoderd(i,y,v);
input [3:0]i;
output [1:0]y;
output v;
assign y[0]=(i[3]|(~i[3]&~i[2]&i[1]));
assign y[1]=(i[3]|(~i[3]&i[2]));
assign v=i[3]|i[2]|i[1]|i[0];
endmodule
```



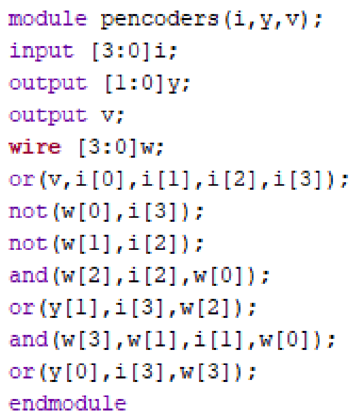
Name	Val
i[3:0]	8
[3]	1
[2]	0
[1]	0
[0]	0
y[1:0]	3
[1]	1
[0]	1

b. BEHAVI



```
module pencoderb(i,y,v);
input [3:0]i;
output reg[1:0]y;
output reg v;
always @(*)
begin
if(i==4'b0000)
begin
y=2'b00;
v=0;
end
else if(i==4'b0001)
begin
y=2'b00;
v=1;
end
else if(i>4'b0001 && i<=4'b0011)
begin
y=2'b01;
v=1;
end
else if(i>4'b0011 && i<=4'b0111)
begin
y=2'b10;
v=1;
end
else if(i>4'b0111 && i<=4'b1111)
begin
y=2'b11;
v=1;
end
end
endmodule
```

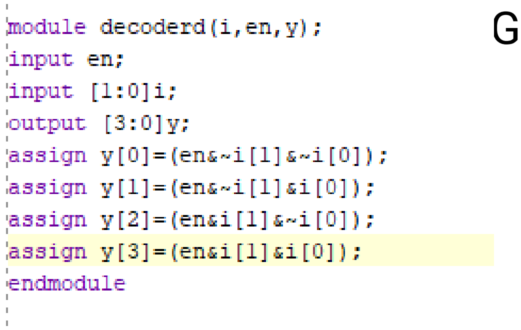
c. STRUCTUR



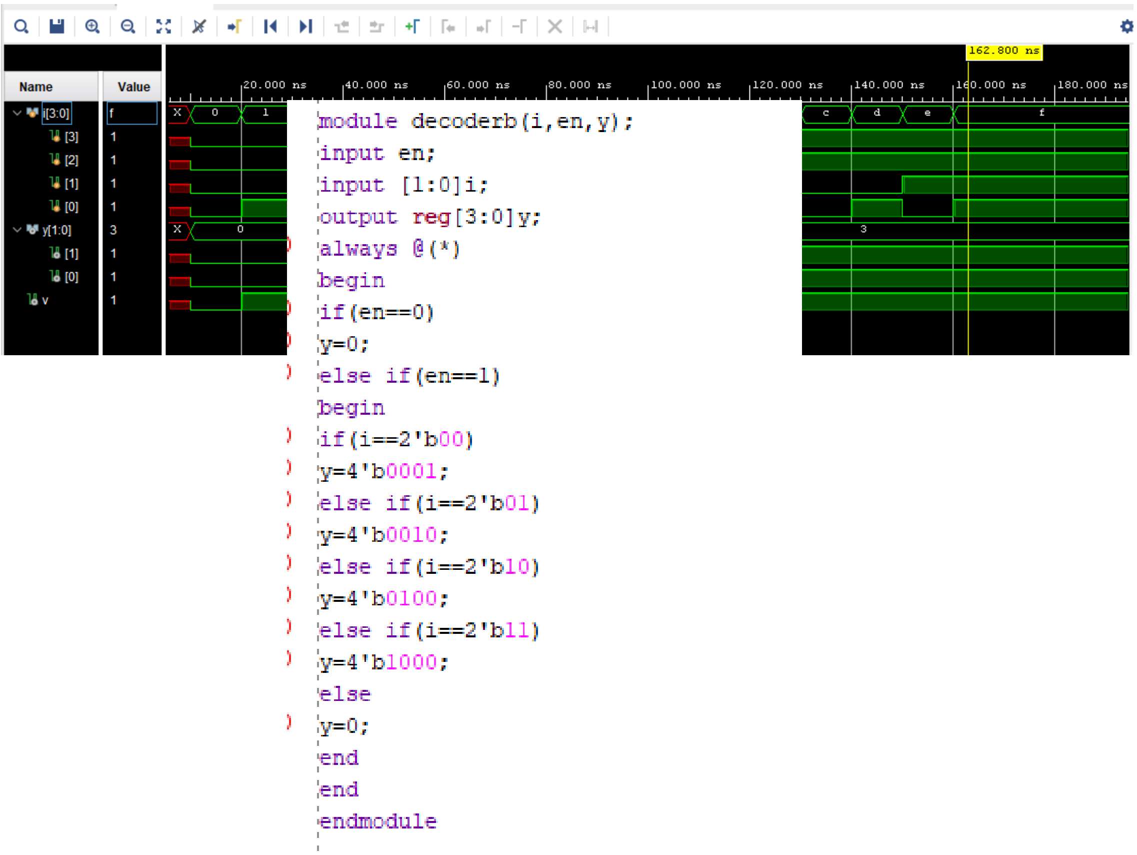
```
module pencoders(i,y,v);
input [3:0]i;
output [1:0]y;
output v;
wire [3:0]w;
or(v,i[0],i[1],i[2],i[3]);
not(w[0],i[3]);
not(w[1],i[2]);
and(w[2],i[2],w[0]);
or(y[1],i[3],w[2]);
and(w[3],w[1],i[1],w[0]);
or(y[0],i[3],w[3]);
endmodule
```

6. DECODERS

a. 2:4 DECOD G)

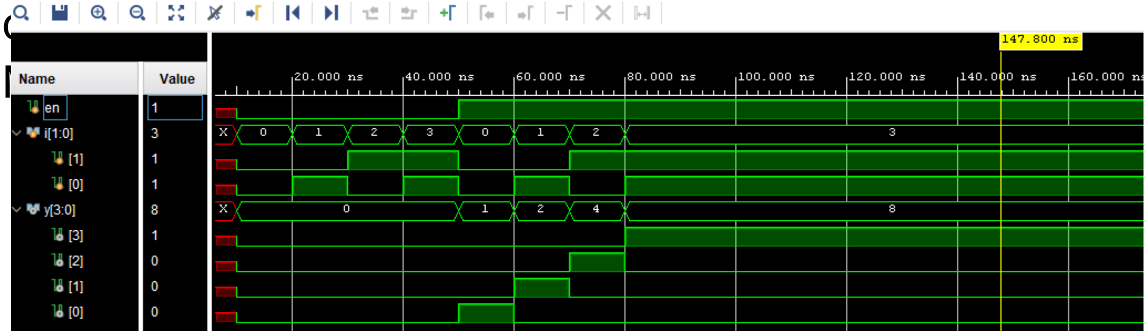


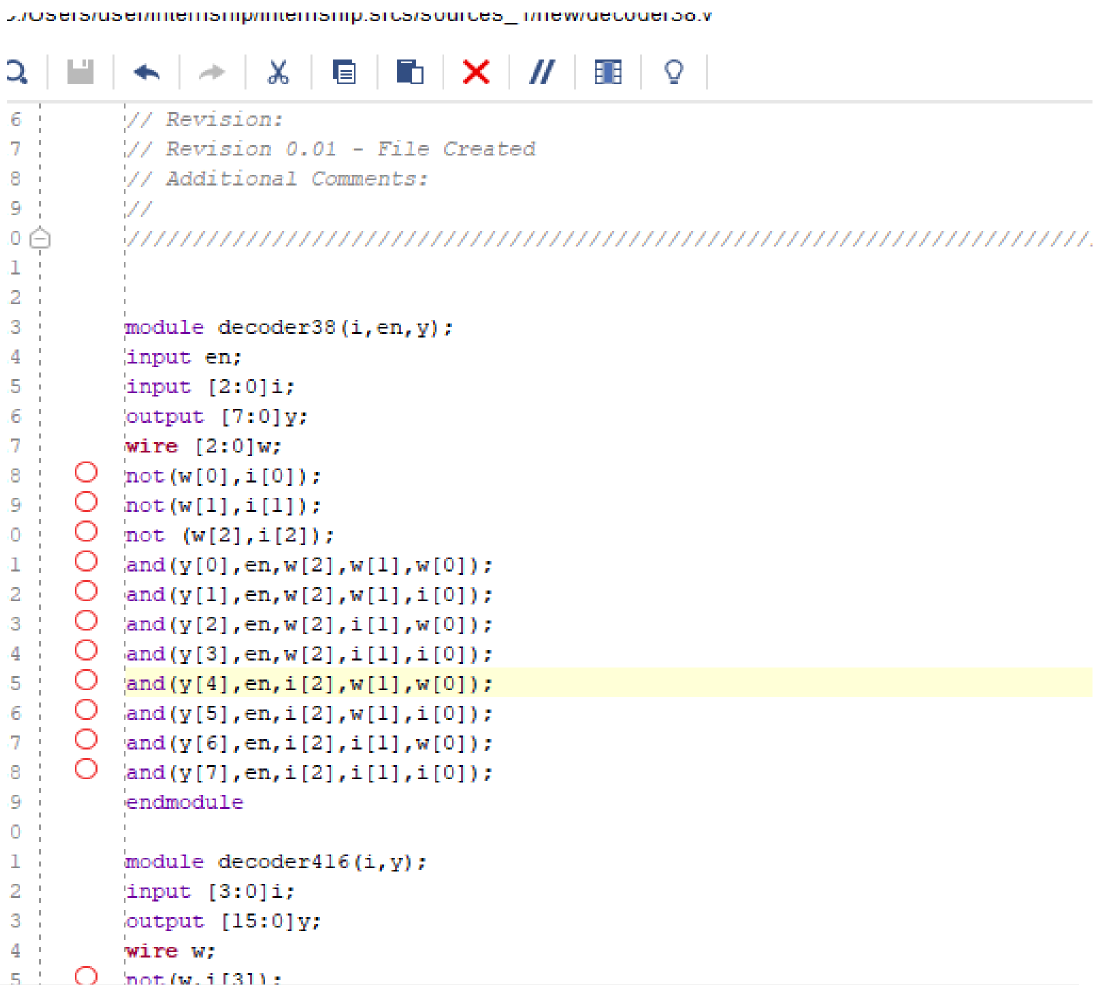
```
module decoderd(i,en,y);
input en;
input [1:0]i;
output [3:0]y;
assign y[0]=(en&~i[1]&~i[0]);
assign y[1]=(en&~i[1]&i[0]);
assign y[2]=(en&i[1]&~i[0]);
assign y[3]=(en&i[1]&i[0]);
endmodule
```



Name	Value
i[3:0]	f
[3]	1
[2]	1
[1]	1
[0]	1
y[1:0]	3
[1]	1
[0]	1
v	1

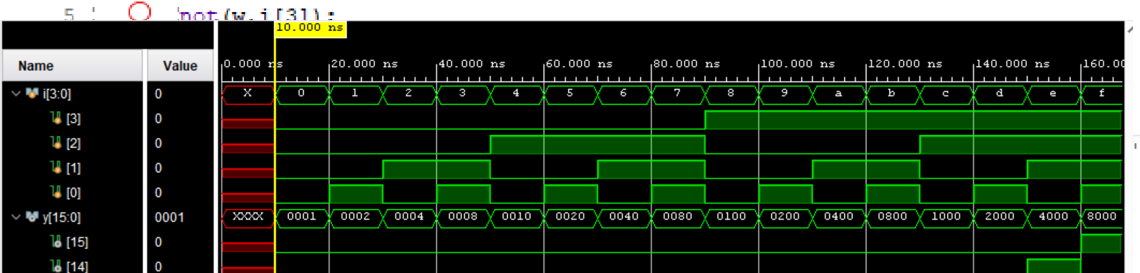
```
module decoderb(i,en,y);
input en;
input [1:0]i;
output reg[3:0]y;
always @(*)
begin
if(en==0)
y=0;
else if(en==1)
begin
if(i==2'b00)
y=4'b0001;
else if(i==2'b01)
y=4'b0010;
else if(i==2'b10)
y=4'b0100;
else if(i==2'b11)
y=4'b1000;
else
y=0;
end
end
endmodule
```



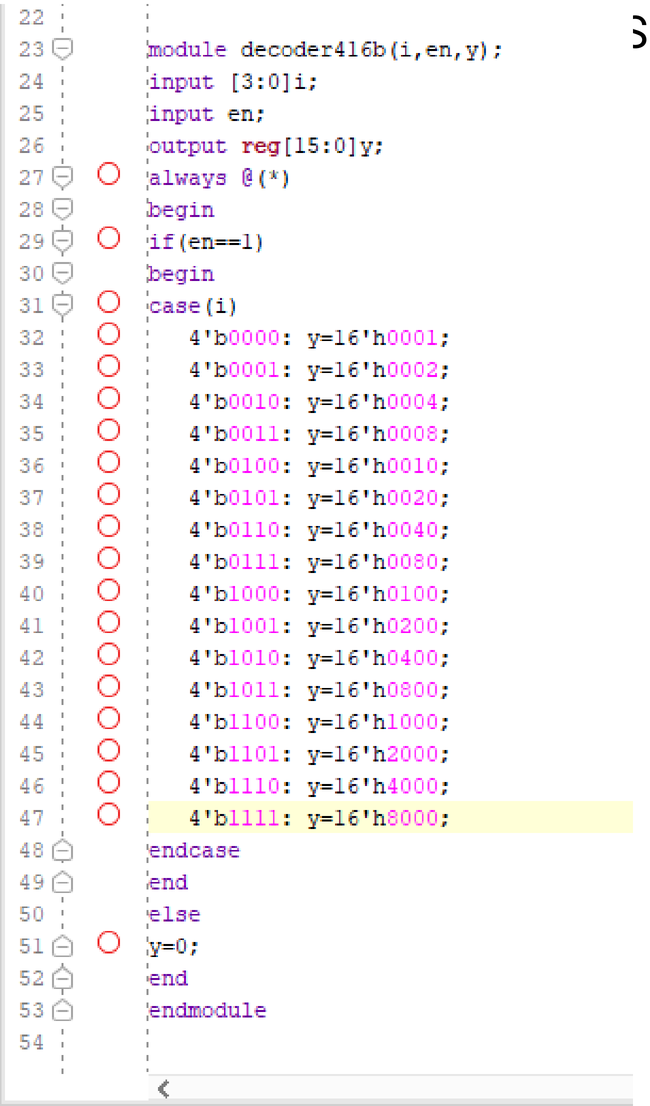


s:/Users/user/internship/internship.srcs/sources_1/new/decoder38.v

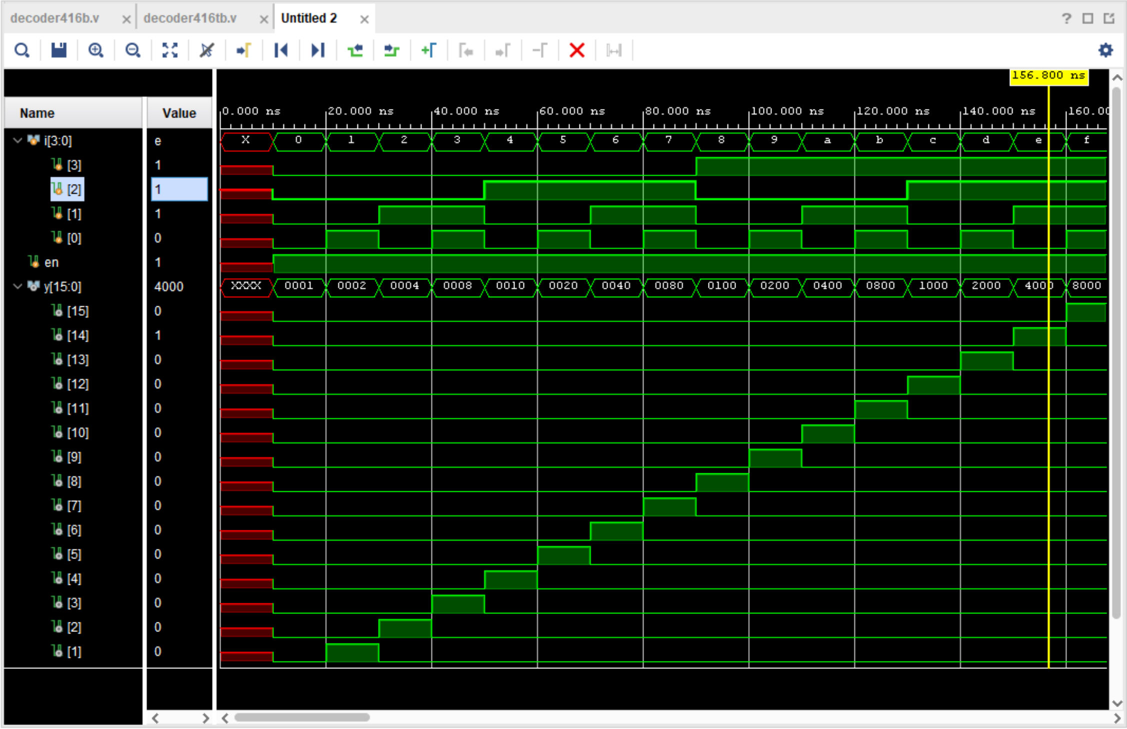
```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////////////////////////


module decoder38(i,en,y);
input en;
input [2:0]i;
output [7:0]y;
wire [2:0]w;
not(w[0],i[0]);
not(w[1],i[1]);
not (w[2],i[2]);
and(y[0],en,w[2],w[1],w[0]);
and(y[1],en,w[2],w[1],i[0]);
and(y[2],en,w[2],i[1],w[0]);
and(y[3],en,w[2],i[1],i[0]);
and(y[4],en,i[2],w[1],w[0]);
and(y[5],en,i[2],w[1],i[0]);
and(y[6],en,i[2],i[1],w[0]);
and(y[7],en,i[2],i[1],i[0]);
endmodule

module decoder416(i,y);
input [3:0]i;
output [15:0]y;
wire w;
not(w,i[3]);
```

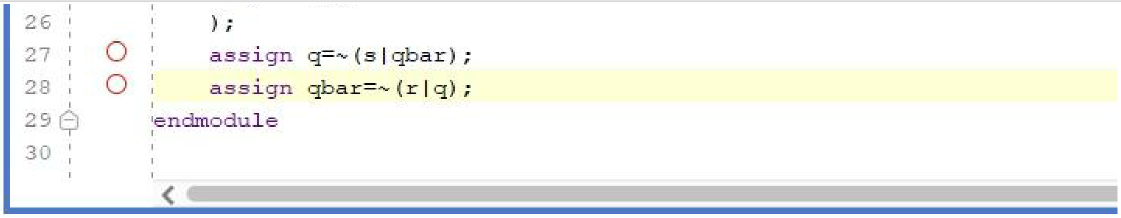


d. 4:16 DECC S



```
module decoder416b(i,en,y);
input [3:0]i;
input en;
output reg[15:0]y;
always @(*)
begin
if(en==1)
begin
case(i)
   4'b0000: y=16'h0001;
   4'b0001: y=16'h0002;
   4'b0010: y=16'h0004;
   4'b0011: y=16'h0008;
   4'b0100: y=16'h0010;
   4'b0101: y=16'h0020;
   4'b0110: y=16'h0040;
   4'b0111: y=16'h0080;
   4'b1000: y=16'h0100;
   4'b1001: y=16'h0200;
   4'b1010: y=16'h0400;
   4'b1011: y=16'h0800;
   4'b1100: y=16'h1000;
   4'b1101: y=16'h2000;
   4'b1110: y=16'h4000;
   4'b1111: y=16'h8000;
endcase
end
else
y=0;
end
endmodule
```

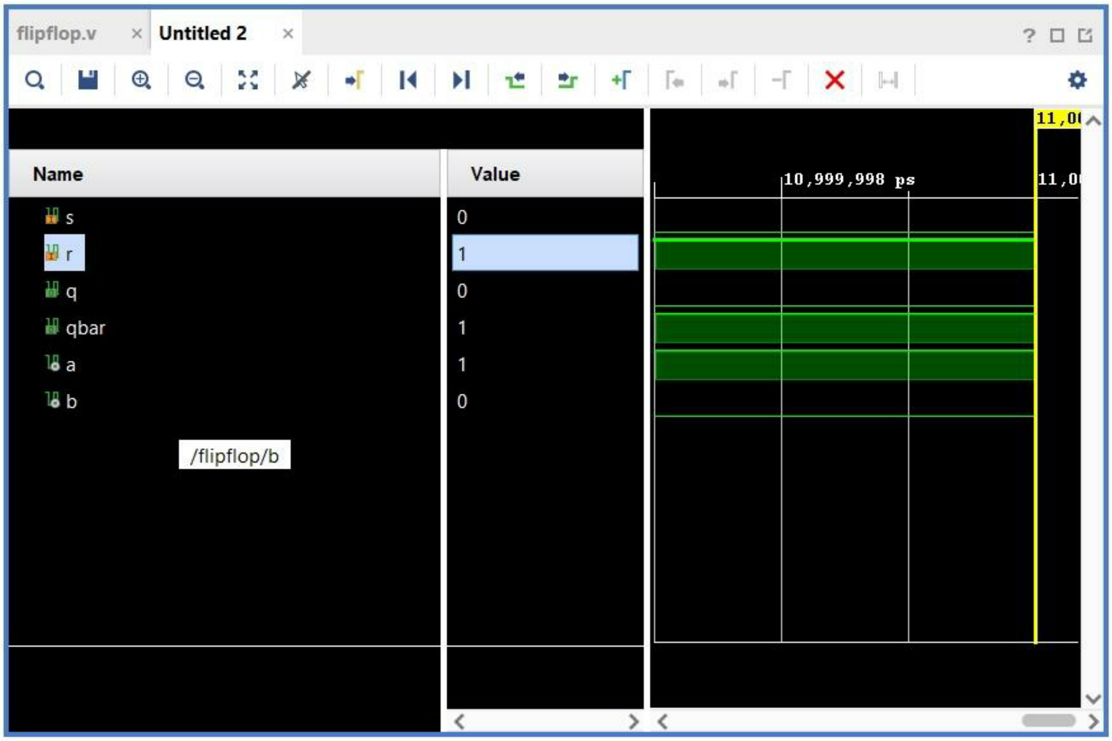




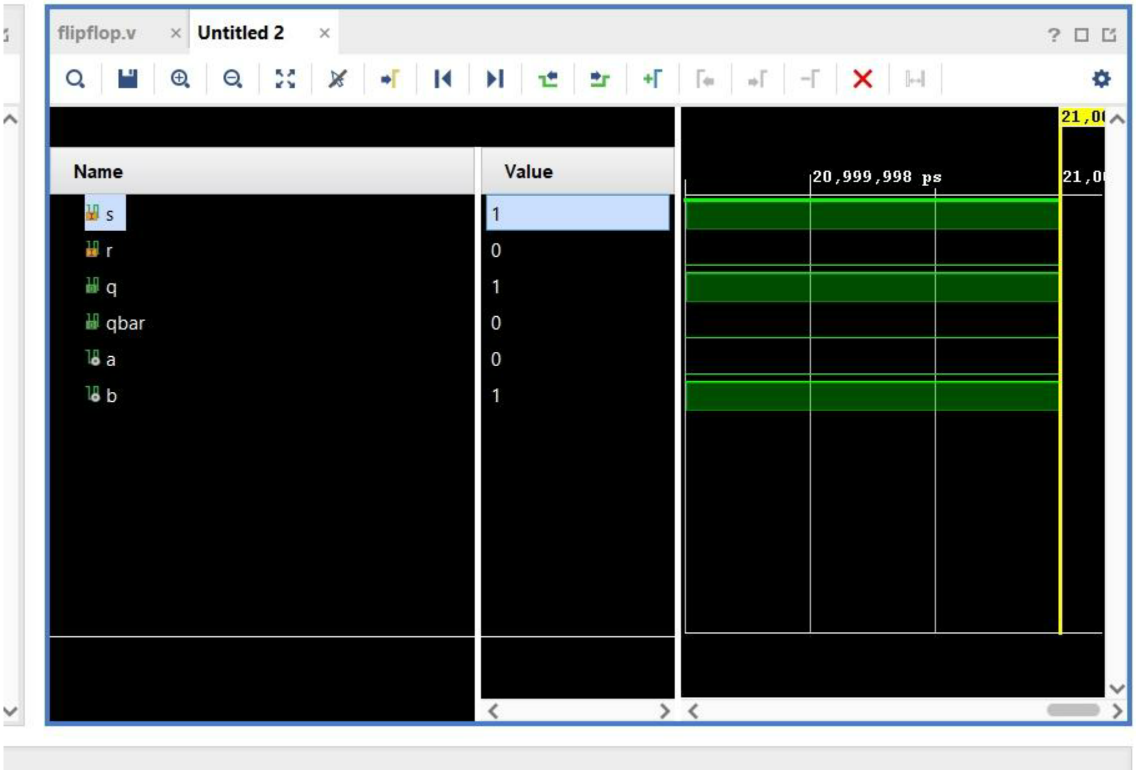
```
    );
    assign q=~(s|qbar);
    assign qbar=~(r|q);
endmodule
```

7.FLIPFLOPS

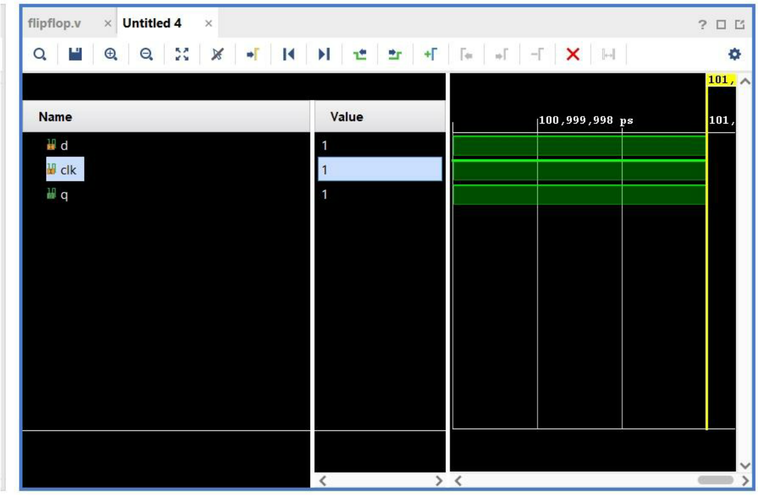
a.SR Flipflops



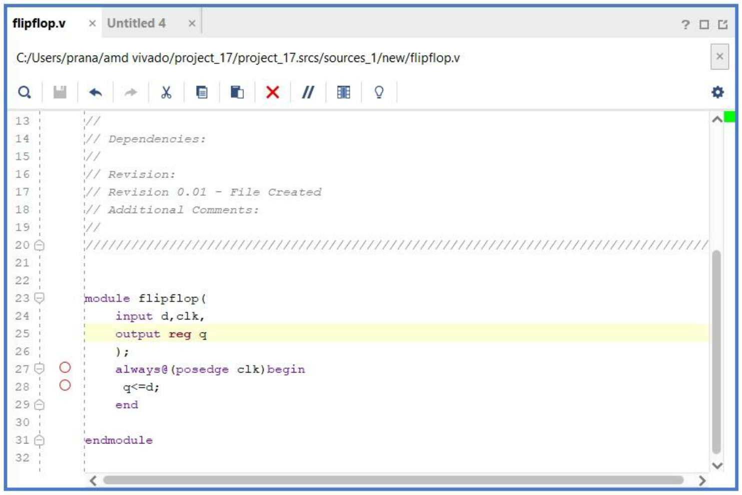
Name	Value
s	0
r	1
q	0
qbar	1
a	1
b	0



Name	Value
s	1
r	0
q	1
qbar	0
a	0
b	1



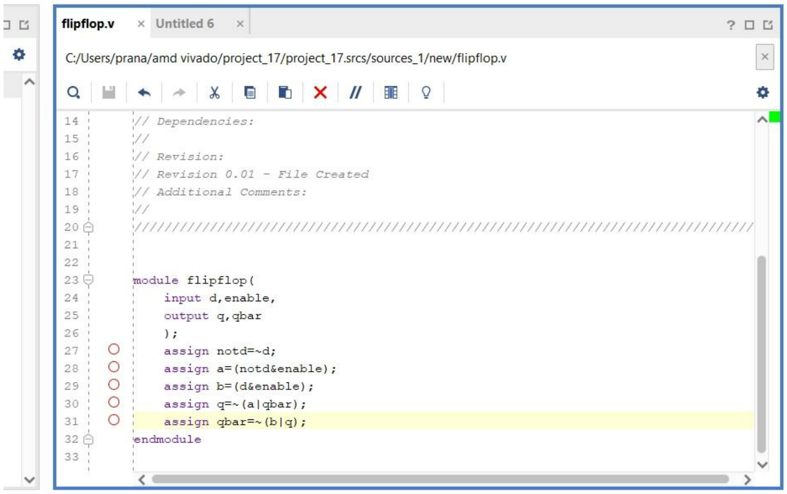
Name	Value
d	1
clk	1
q	1



C:/Users/prana/amd vivado/project_17/project_17.srcs/sources_1/new/flipflop.v

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module flipflop(
    input d,clk,
    output reg q
    );
    always@(posedge clk)begin
     q<=d;
    end

endmodule

```

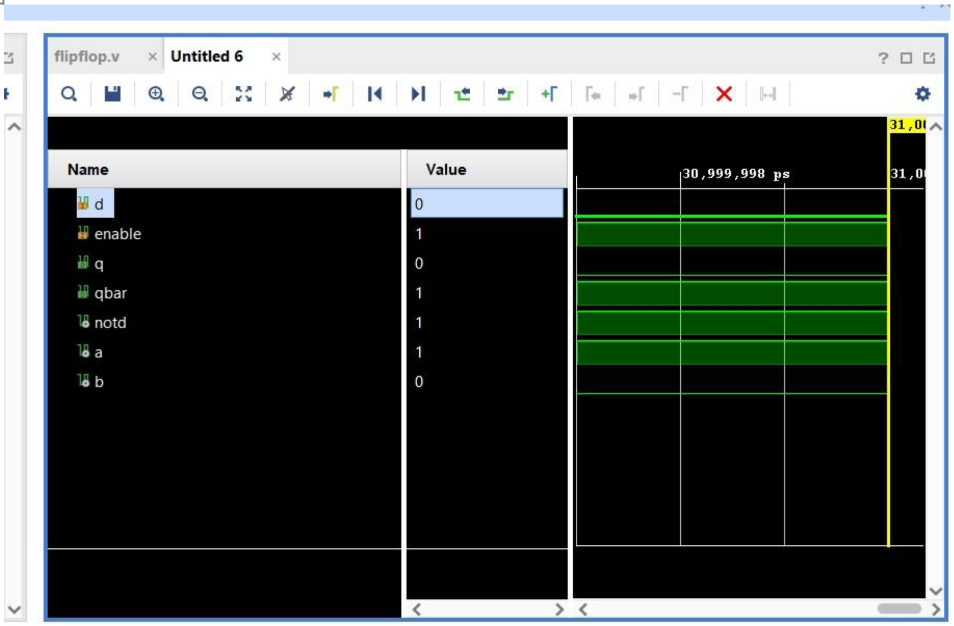
b.D flipflop



C:/Users/prana/amd vivado/project_17/project_17.srcs/sources_1/new/flipflop.v

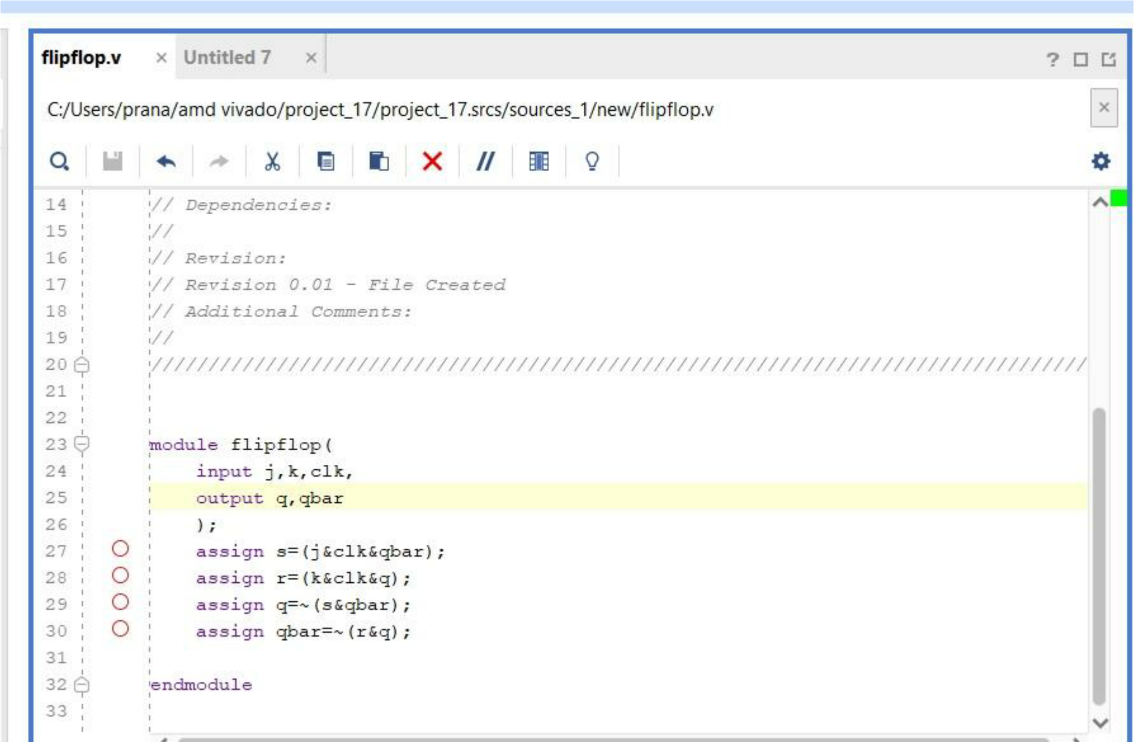
```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module flipflop(
    input d,enable,
    output q,qbar
    );
    assign notd=~d;
    assign a=(notd&enable);
    assign b=(d&enable);
    assign q=~(a|qbar);
    assign qbar=~(b|q);
endmodule

```

c.dlatch



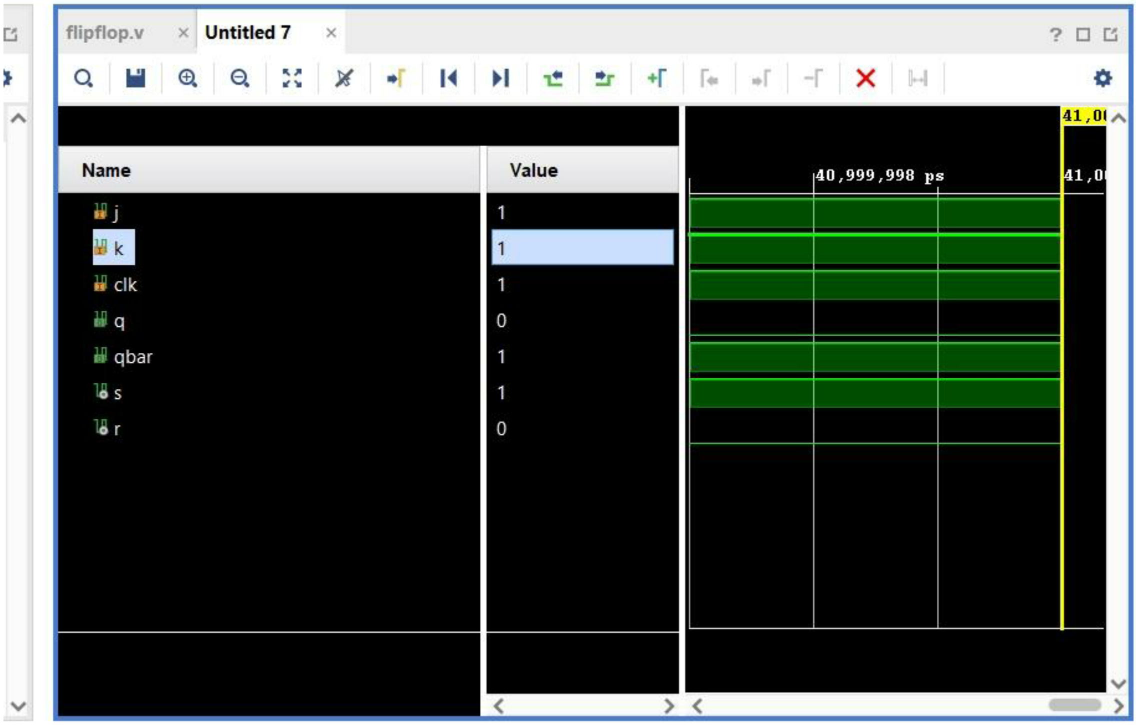
Name	Value
d	0
enable	1
q	0
qbar	1
notd	1
a	1
b	0

d.JK flipflop

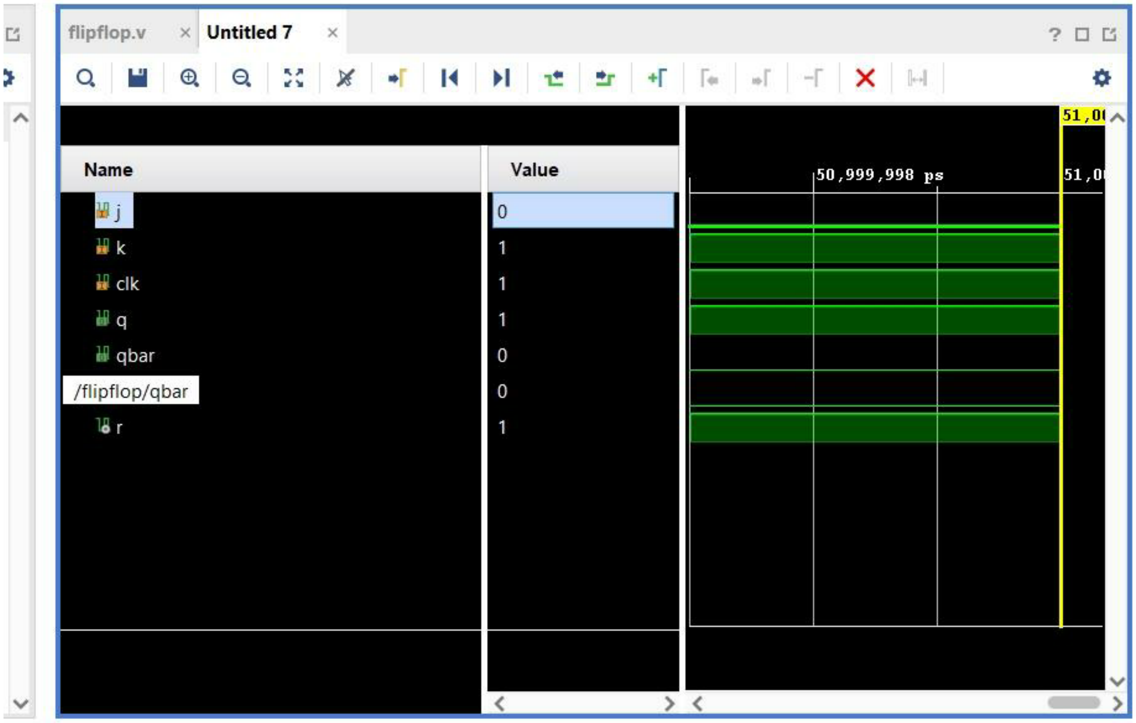


C:/Users/prana/amd vivado/project_17/project_17.srcs/sources_1/new/flipflop.v

```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module flipflop(
    input j,k,clk,
    output q,qbar
    );
    assign s=(j&clk&qbar);
    assign r=(k&clk&q);
    assign q=~(s&qbar);
    assign qbar=~(r&q);

endmodule

```



Name	Value
j	1
k	1
clk	1
q	0
qbar	1
s	1
r	0



Name	Value
j	0
k	1
clk	1
q	1
qbar	0
/flipflop/qbar	0
r	1