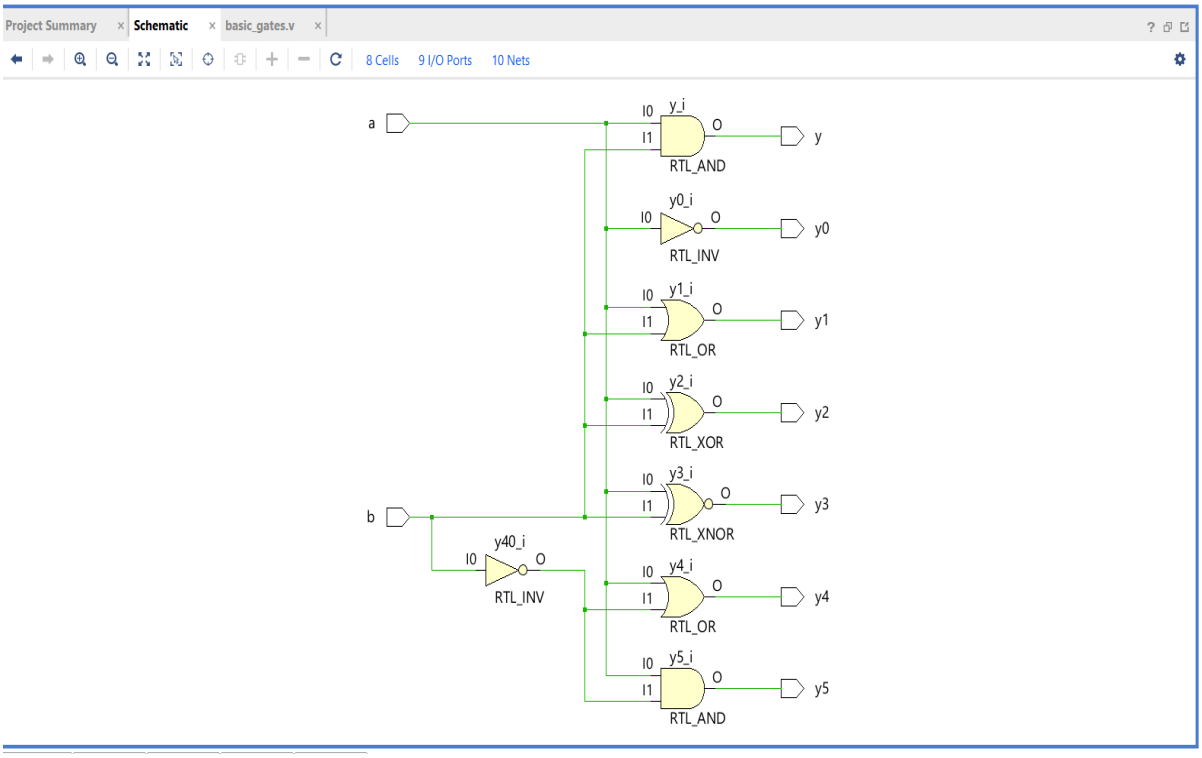
DSD ASSIGNMENT =1

1) BASIC LOGIC GATES

Project Summary | basic_gates.v

C:/Users/vinit/project_2/project_2.srcs/sources_1/new/basic_gates.v

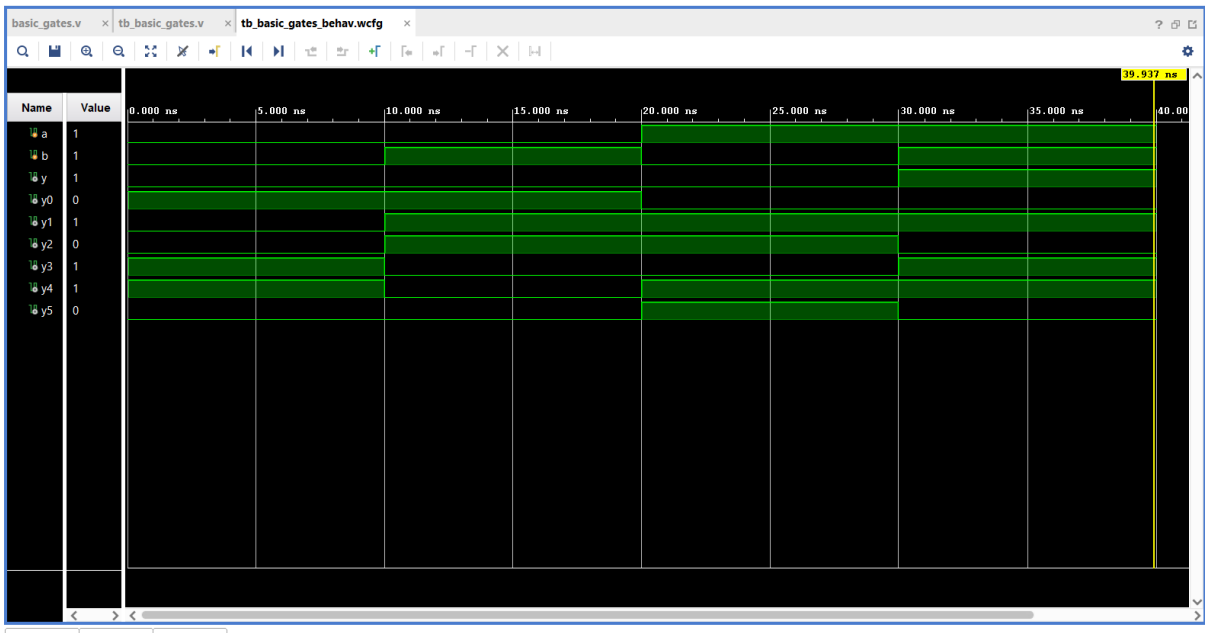
```
1  `timescale 1ns / 1ps
2
3
4
5  module basic_gates(
6  input a,b,
7  output y,y0,y1,y2,y3,y4,y5
8      );
9      assign y= a&b;//and
10     assign y0=~a;//not
11     assign y1= a|b;//or
12       assign y2= a^b;//xor
13        assign y3=a^~b;//xnor
14        assign y4= a|~b;//nor
15        assign y5=a&~b;//nand
16 endmodule
17
```



Project Summary | Schematic | basic_gates.v | tb_basic_gates.v *

C:/Users/vinit/project_2/project_2.srcs/sim_1/new/tb_basic_gates.v

```
1  `timescale 1ns / 1ps
2
3
4
5  module tb_basic_gates(
6
7      );
8      reg a,b;
9      wire y,y0,y1,y2,y3,y4,y5;
10
11     basic_gates uut(a,b,y,y0,y1,y2,y3,y4,y5);
12
13     initial
14     begin
15
16     a=0;b=0;
17     #10
18      a=0;b=1;
19     #10
20      a=1;b=0;
21     #10
22     a=1;b=1;
23     #10
24     $finish;
25     end
26 endmodule
27
```

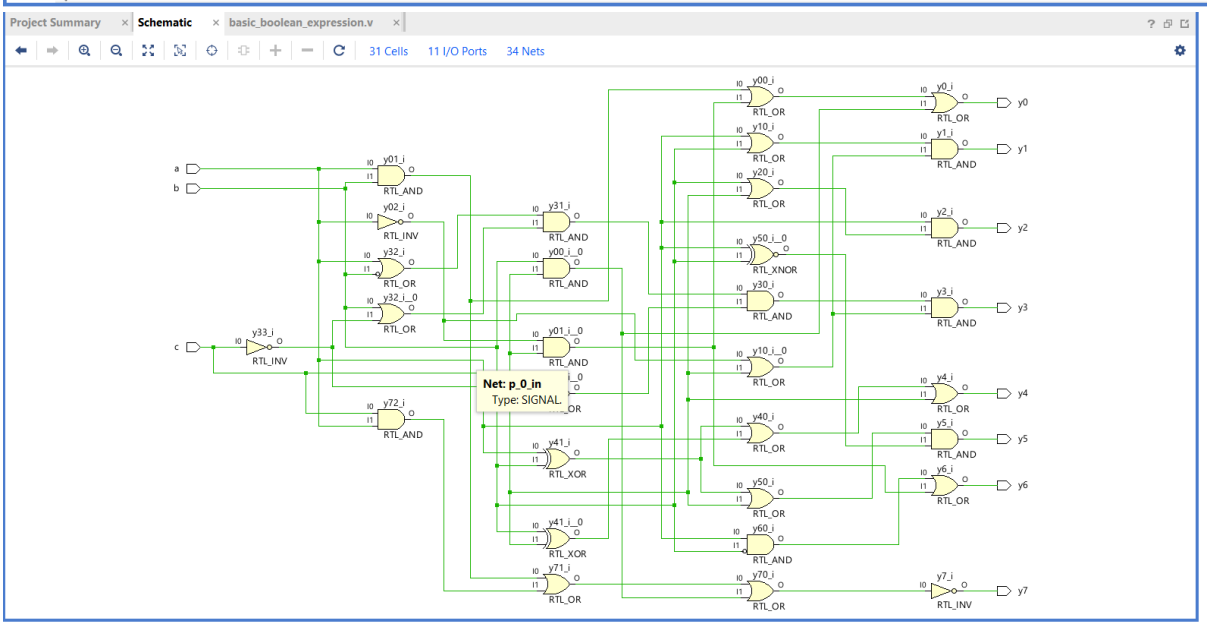


2) BOOLEAN EXPRESSIONS

Project Summary | basic_boolean_expression.v

C:/Users/vinit/project_3/project_3.srcs/sources_1/new/basic_boolean_expression.v

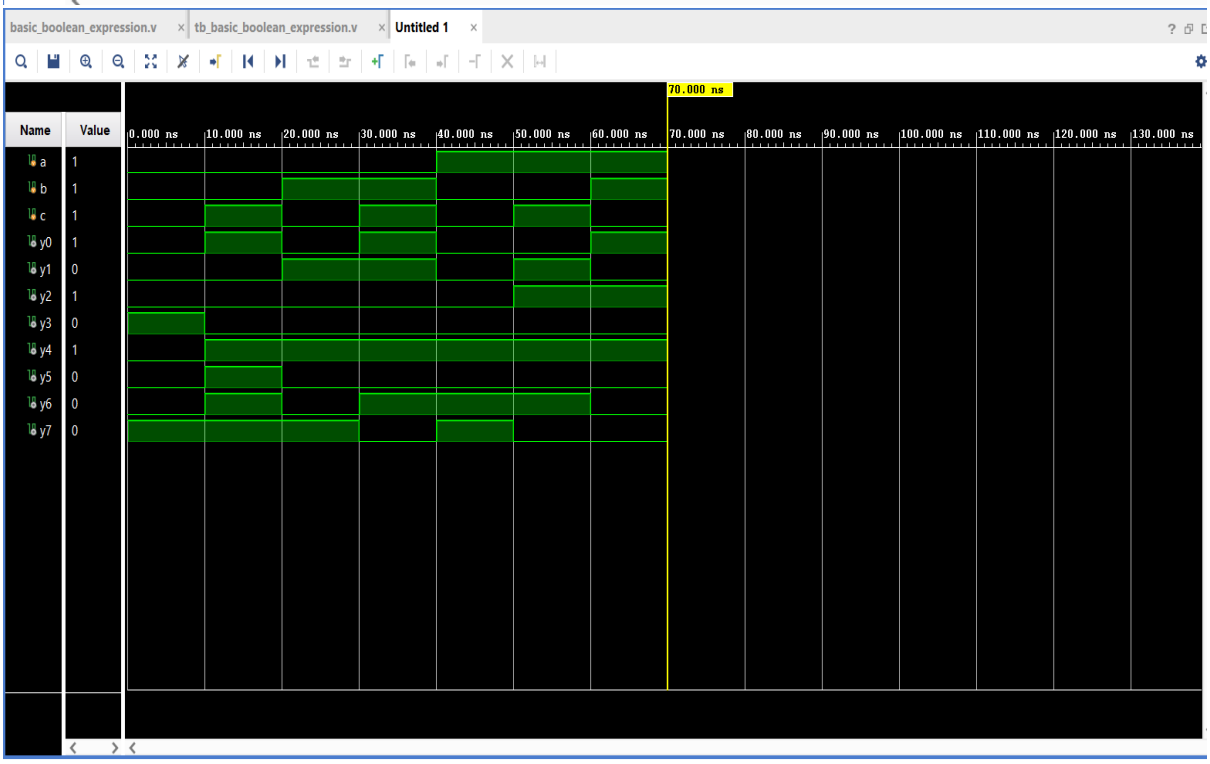
```
1  `timescale 1ns / 1ps
2
3
4  module basic_boolean_expression(
5  input a,b,c,
6  output y0,y1,y2,y3,y4,y5,y6,y7
7      );
8
9   assign y0 =(a&b)|(~a&c)|(b&c);
10  assign y1=(a|b )&  ((~a)|c);
11  assign y2= a&(b|c);
12  assign y3=(a|(~b))&(b|(~c))&((~c)|a)&((~a)|c);
13  assign y4 = (a^b)|(b^c)|c;
14  assign y5 =((a^b)|c)& (a^~b);
15  assign y6=(a&~b)|(c&~a);
16  assign y7 =~((a&b)|(c&a)|(b&c));
17
18 endmodule
```



Project Summary | Schematic | basic_boolean_expression.v | tb_basic_boolean_expression.v *

C:/Users/vinit/project_3/project_3.srcs/sim_1/new/tb_basic_boolean_expression.v

```
3  module tb_basic_boolean_expression(
4      );
5      reg a,b,c;
6      wire y0,y1,y2,y3,y4,y5,y6,y7;
7
8  basic_boolean_expression uut(a,b,c,y0,y1,y2,y3,y4,y5,y6,y7);
9
10     initial
11     begin
12     a =0;    b=0;    c=0;
13     #10
14      a =0;   b=0;    c=1;
15     #10
16      a =0;    b=1;    c=0;
17     #10
18      a =0;    b=1;    c=1;
19      #10
20      a =1;    b=0;    c=0;
21     #10
22      a =1;    b=0;    c=1;
23     #10
24      a =1;    b=1;    c=0;
25     #10
26     a =1;    b=1;    c=1;
27     $finish;
28     end
29 endmodule
30
```

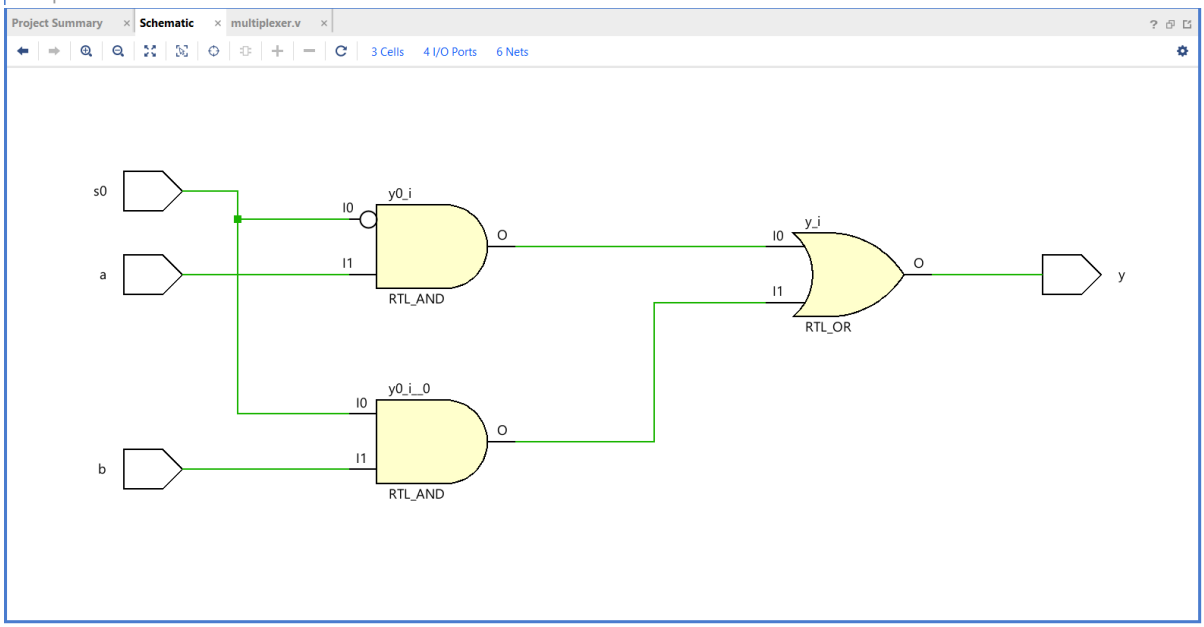


3a) MUX 2X1

Project Summary × multiplexer.v ×

C:/Users/vinit/project_4/project_4.srcs/sources_1/new/multiplexer.v

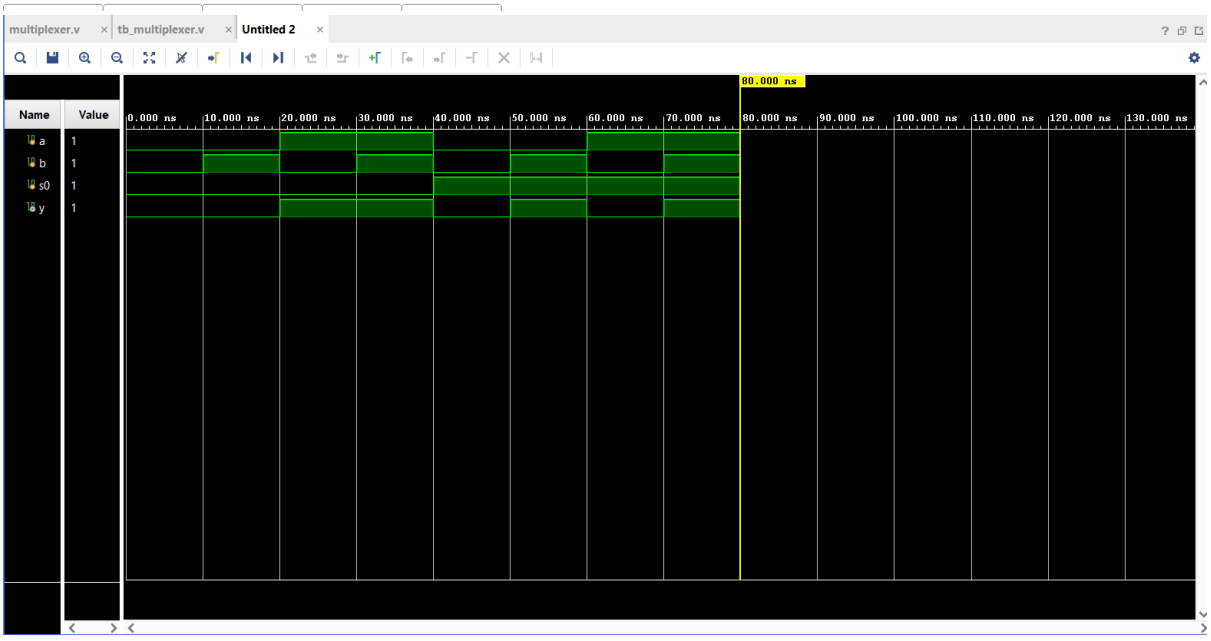
```
1  `timescale 1ns / 1ps
2
3  module multiplexer(
4  input a,b,s0,
5  output y
6      );
7      assign y =(~s0&a)|(s0&b);
8  endmodule
9
```



Project Summary | Schematic | multiplexer.v | tb_multiplexer.v *

C:/Users/vinit/project_4/project_4.srcs/sim_1/new/tb_multiplexer.v

```
1  `timescale 1ns / 1ps
2  module tb_multiplexer(
3      );
4      reg a,b,s0;
5      wire y;
6
7
8      multiplexer uut (a,b,s0,y);
9
10     initial
11     begin
12     a=0;b=0;s0=0;
13     #10
14     a=0;b=1;s0=0;
15     #10
16     a=1;b=0;s0=0;
17     #10
18     a=1;b=1;s0=0;
19     #10
20      a=0;b=0;s0=1;
21     #10
22     a=0;b=1;s0=1;
23     #10
24     a=1;b=0;s0=1;
25     #10
26     a=1;b=1;s0=1;
27     #10
28     $finish;
29     end
30 endmodule
31
```

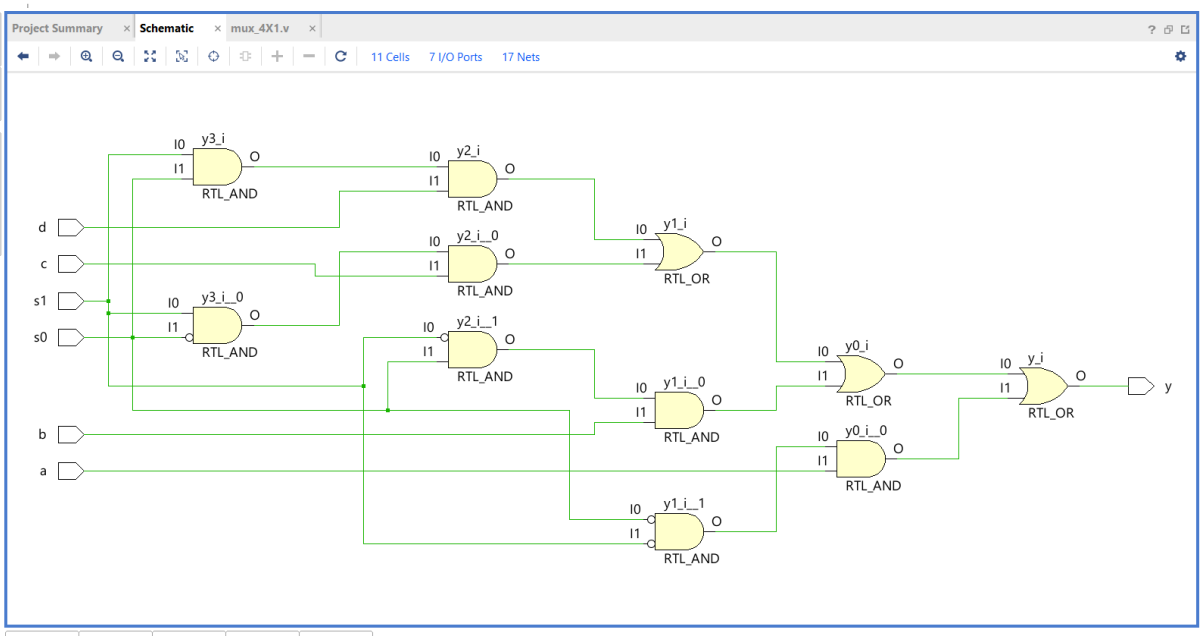


3b) MUX 4X1

Project Summary | mux_4X1.v

C:/Users/vinit/project_4X1_mux/project_4X1_mux.srcs/sources_1/new/mux_4X1.v

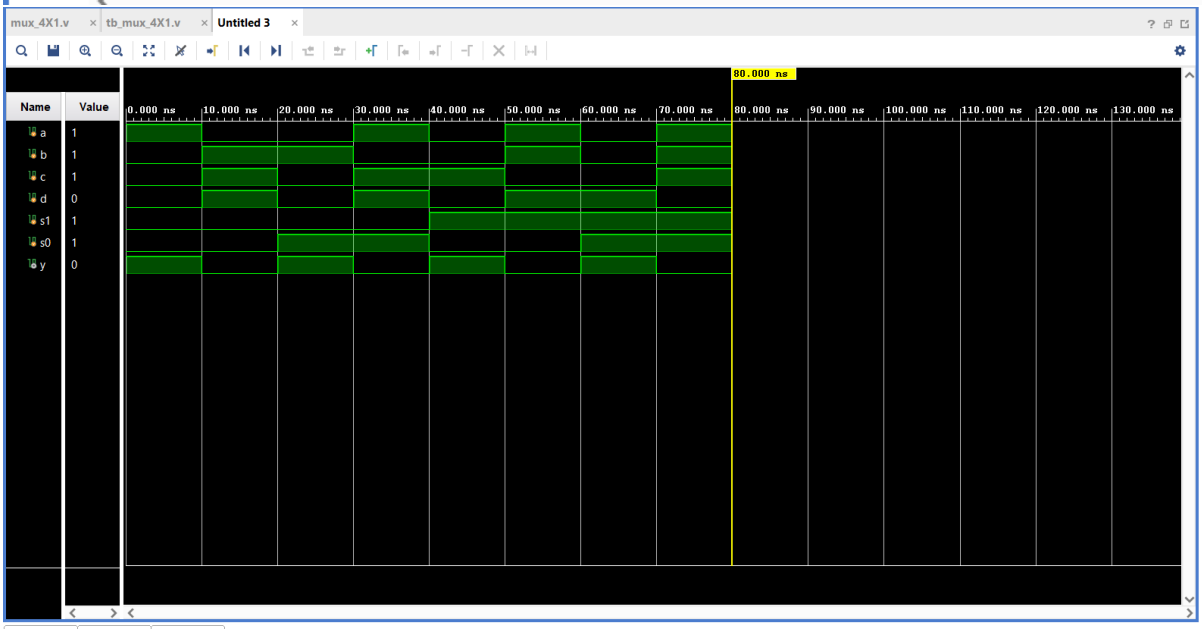
```
1  `timescale 1ns / 1ps
2  module mux_4X1(
3  input a,b,c,d,s0,s1 ,output y
4      );
5      assign y=(s1&s0&d)|(s1&(~s0)&c)|((~s1)&s0&b)|((~s0)&(~s1)&a);
6  endmodule
7
```



Project Summary | Schematic | mux_4X1.v | tb_mux_4X1.v *

C:/Users/vinit/project_4X1_mux/project_4X1_mux.srcs/sim_1/new/tb_mux_4X1.v

```
1  `timescale 1ns / 1ps
2  module tb_mux_4X1(
3      );
4        reg a,b,c,d,s1,s0;
5      wire y;
6      mux_4X1 uut(a,b,c,d,s0,s1,y);
7      initial
8      begin
9    a=1; b=0; c=0; d=0; s0=0; s1=0;
10   #10;
11   a=0; b=1; c=1; d=1; s0=0; s1=0;
12   #10;
13   a=0; b=1; c=0; d=0; s0=1; s1=0;
14   #10;
15   a=1; b=0; c=1; d=1; s0=1; s1=0;
16   #10;
17   a=0; b=0; c=1; d=0; s0=0; s1=1;
18   #10;
19   a=1; b=1; c=0; d=1; s0=0; s1=1;
20   #10;
21   a=0; b=0; c=0; d=1; s0=1; s1=1;
22   #10;
23     a=1; b=1; c=1; d=0; s0=1; s1=1;
24   #10;
25     $finish;
26     end
27
28 endmodule
29
```

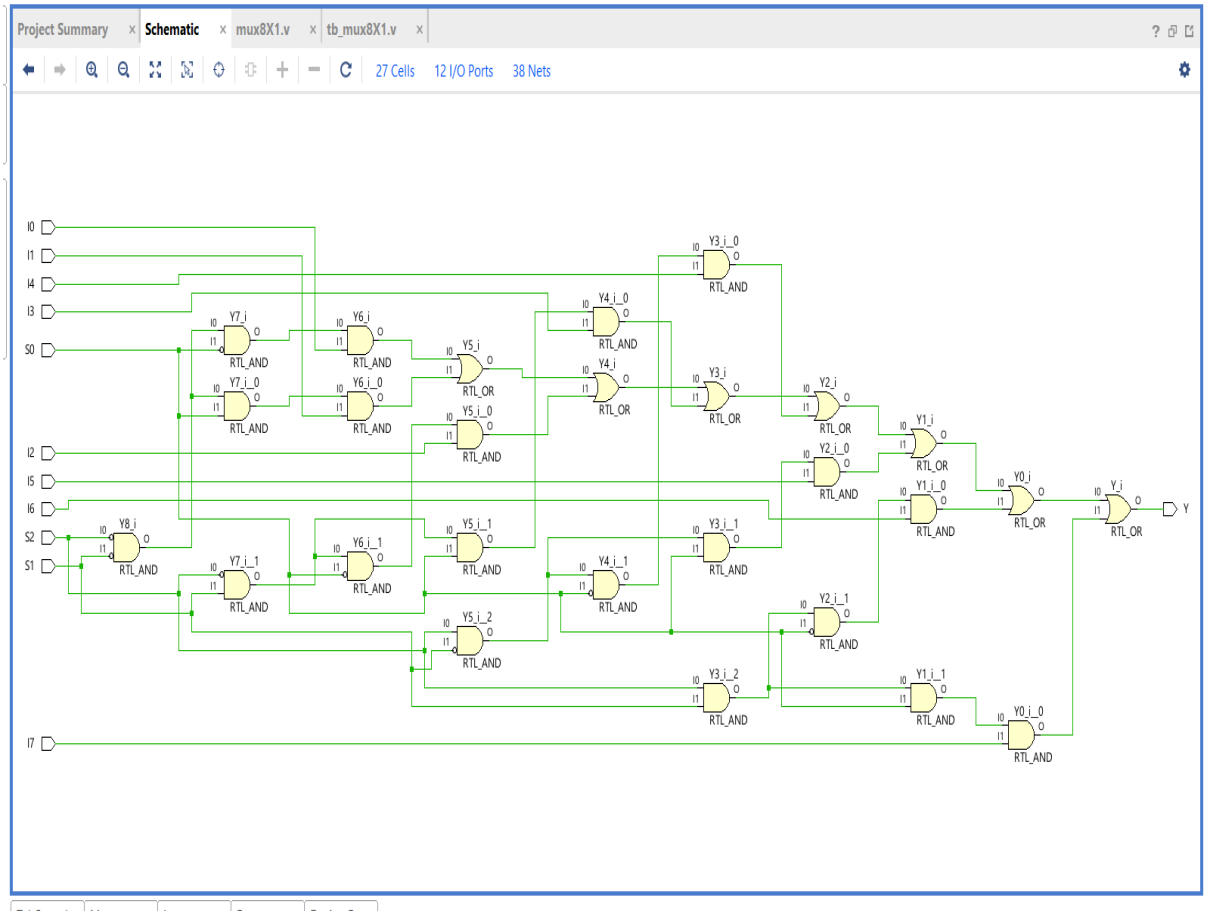


3c) MUX 8X1

Project Summary | mux8X1.v *

C:/Users/vinit/project_mux8X1/project_mux8X1.srcs/sources_1/new/mux8X1.v

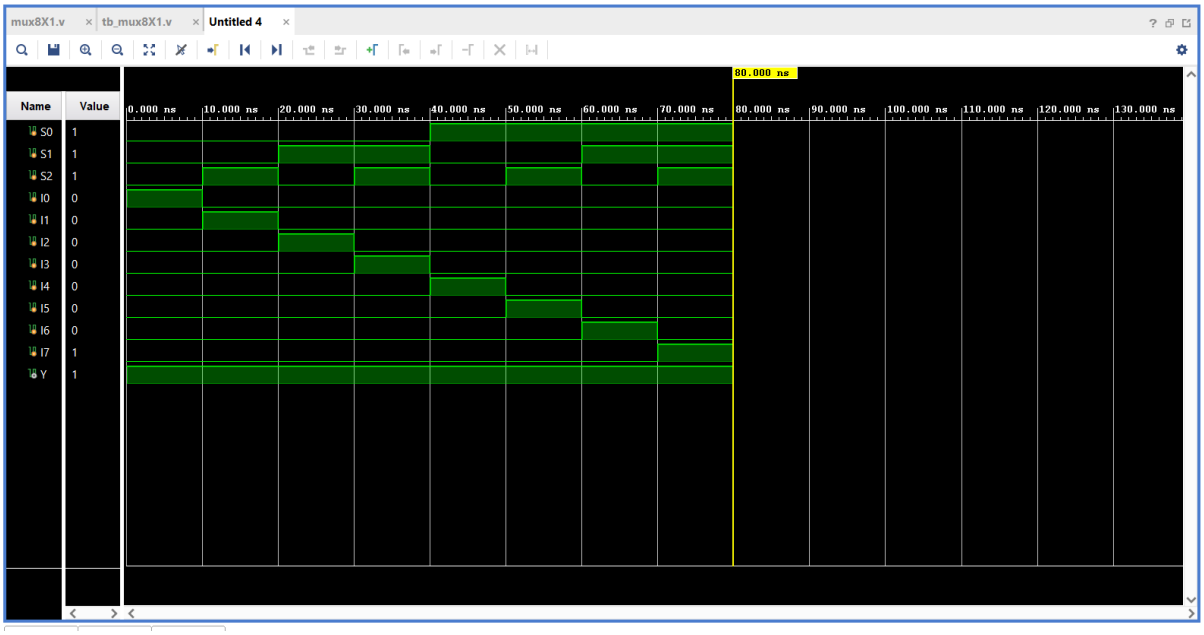
```
1  `timescale 1ns / 1ps
2
3  module mux8X1(
4  input S0,S1,S2,I0,I1,I2,I3,I4,I5,I6,I7, output Y
5      );
6      assign Y = (~S2 & ~S1 & ~S0 & I0)|(~S2 & ~S1 &  S0 & I1)|
7      (~S2 &  S1 & ~S0 & I2)|(~S2 &  S1 &  S0 & I3)|
8      ( S2 & ~S1 & ~S0 & I4)|( S2 & ~S1 &  S0 & I5)|
9      ( S2 &  S1 & ~S0 & I6)|( S2 &  S1 &  S0 & I7);
10 endmodule
11
```



Project Summary | mux8X1.v * | tb_mux8X1.v

C:/Users/vinit/project_mux8X1/project_mux8X1.srcs/sim_1/new/tb_mux8X1.v

```
1  `timescale 1ns / 1ps
2
3  module tb_mux8X1(
4
5      );
6      reg S0,S1,S2,I0,I1,I2,I3,I4,I5,I6,I7;
7      wire Y;
8
9      mux8X1 uut(S2,S1,S0,I0,I1,I2,I3,I4,I5,I6,I7,Y);
10
11     initial
12     begin
13     S0=0;S1=0;S2=0;I0=1;I1=0;I2=0;I3=0;I4=0;I5=0;I6=0;I7=0;
14     #10
15      S0=0;S1=0;S2=1;I0=0;I1=1;I2=0;I3=0;I4=0;I5=0;I6=0;I7=0;
16     #10
17      S0=0;S1=1;S2=0;I0=0;I1=0;I2=1;I3=0;I4=0;I5=0;I6=0;I7=0;
18     #10 S0=0;S1=1;S2=1;I0=0;I1=0;I2=0;I3=1;I4=0;I5=0;I6=0;I7=0;
19     #10
20      S0=1;S1=0;S2=0;I0=0;I1=0;I2=0;I3=0;I4=1;I5=0;I6=0;I7=0;
21     #10
22      S0=1;S1=0;S2=1;I0=0;I1=0;I2=0;I3=0;I4=0;I5=1;I6=0;I7=0;
23     #10
24      S0=1;S1=1;S2=0;I0=0;I1=0;I2=0;I3=0;I4=0;I5=0;I6=1;I7=0;
25     #10
26     S0=1;S1=1;S2=1;I0=0;I1=0;I2=0;I3=0;I4=0;I5=0;I6=0;I7=1;
27     #10
28
29     $finish;
30      end
31 endmodule
```

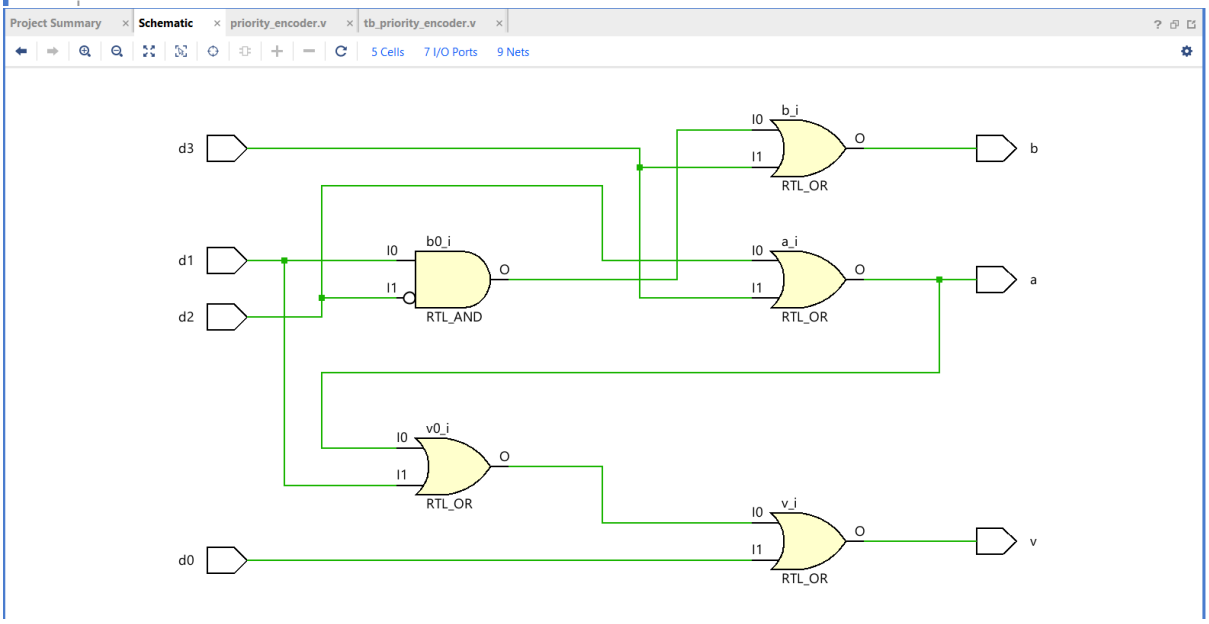


4) PRIORITY ENCODER

Project Summary | priority_encoder.v

C:/Users/vinit/project_5/project_5.srcs/sources_1/new/priority_encoder.v

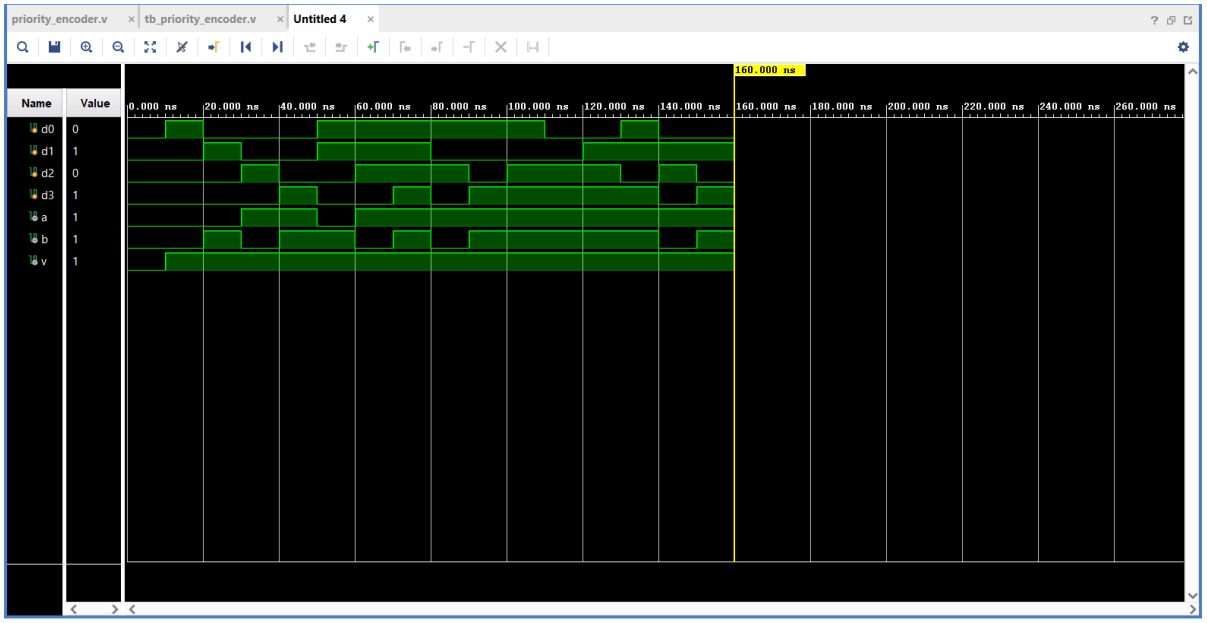
```
1  `timescale 1ns / 1ps
2  module priority_encoder(
3  input d0,d1,d2,d3,
4  output a,b,v
5      );
6      assign a = d2|d3;
7      assign b=( d1 &(~d2))|d3;
8      assign v = d3|d2|d1|d0;
9  endmodule
10
```



Project Summary | priority_encoder.v | tb_priority_encoder.v *

C:/Users/vinit/project_5/project_5.srcs/sim_1/new/tb_priority_encoder.v

```
1  `timescale 1ns / 1ps
2  module tb_priority_encoder(
3      );
4      reg d0,d1,d2,d3;
5      wire a,b ,v;
6          priority_encoder uut(d0,d1,d2,d3,a,b,v);
7          initial
8      begin
9      d0=0; d1=0;d2=0;d3=0;     #10
10     d0=1; d1=0;d2=0;d3=0;     #10
11     d0=0; d1=1;d2=0;d3=0;     #10
12     d0=0; d1=0;d2=1;d3=0;     #10
13     d0=0; d1=0;d2=0;d3=1;     #10
14     d0=1; d1=1;d2=0;d3=0;     #10
15     d0=1; d1=1;d2=1;d3=0;     #10
16     d0=1; d1=1;d2=1;d3=1;     #10
17     d0=1; d1=0;d2=1;d3=0;     #10
18     d0=1; d1=0;d2=0;d3=1;     #10
19     d0=1; d1=0;d2=1;d3=1;     #10
20     d0=0; d1=0;d2=1;d3=1;     #10
21     d0=0; d1=1;d2=1;d3=1;     #10
22     d0=1; d1=1;d2=0;d3=1;     #10
23     d0=0; d1=1;d2=1;d3=0;     #10
24     d0=0; d1=1;d2=0;d3=1;     #10
25
26   $finish;
27     end
28 endmodule
29
```

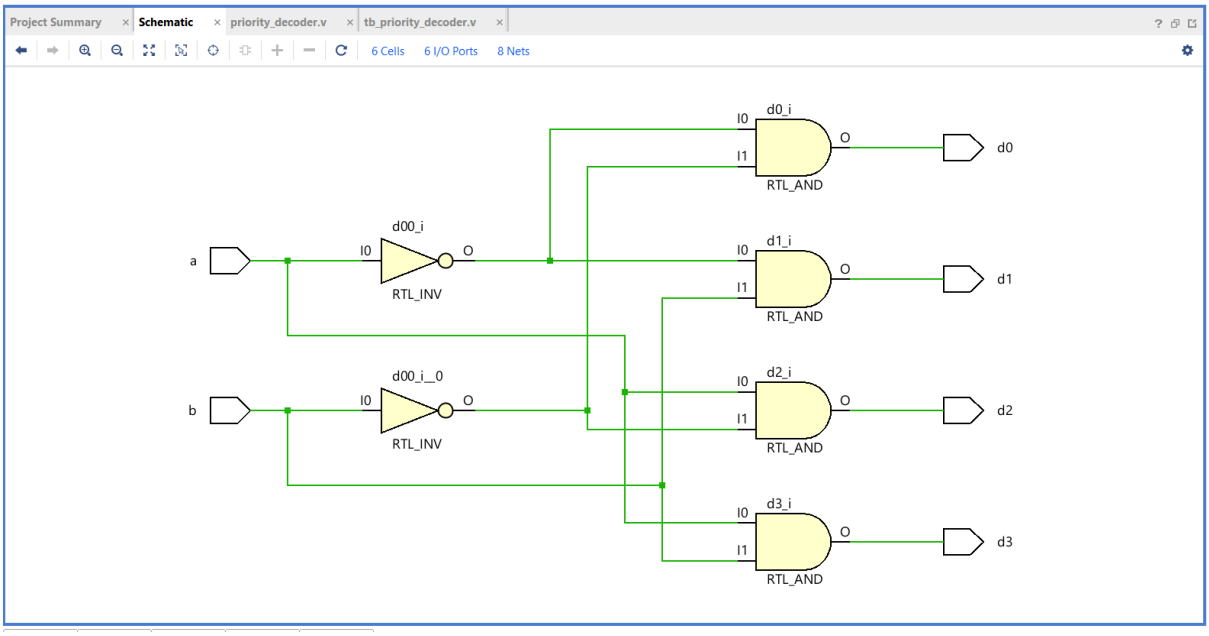


5) PRIORITY DECODER

Project Summary | priority_decoder.v

C:/Users/vinit/project_6/project_6.srcs/sources_1/new/priority_decoder.v

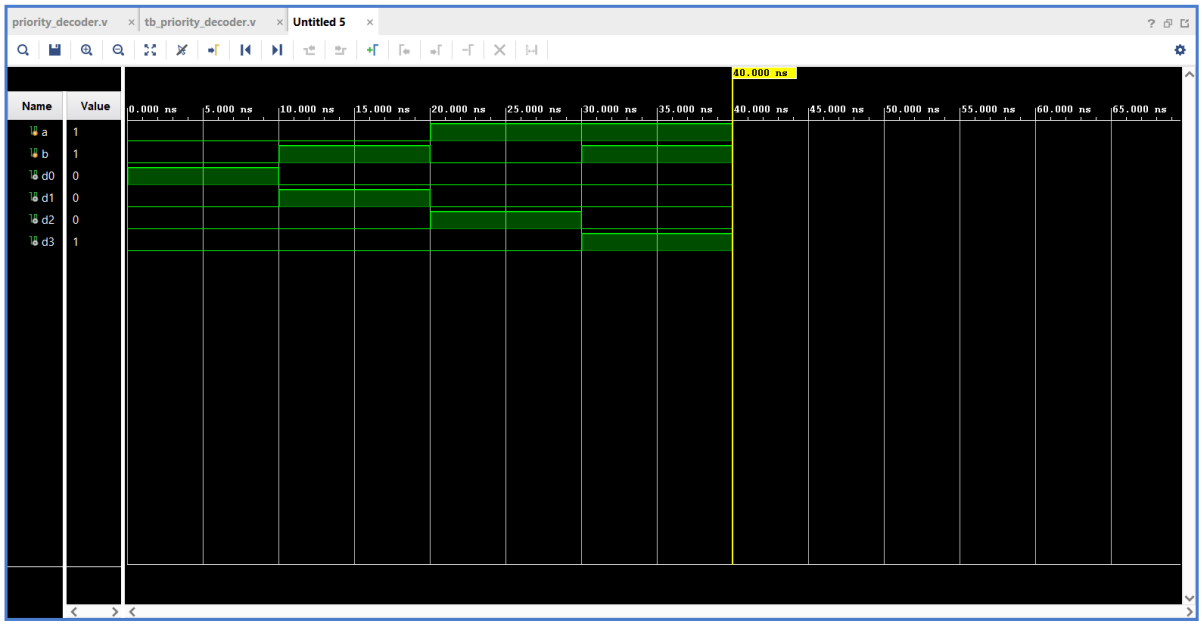
```
1  `timescale 1ns / 1ps
2  module priority_decoder(
3  input a,b,
4  output d0,d1,d2,d3
5      );
6      assign d0 = ~a&(~b);
7      assign d1 = ~a&b;
8      assign d2 = a&(~b);
9      assign d3 = a&b;
10 endmodule
11
```



Project Summary | priority_decoder.v | tb_priority_decoder.v

C:/Users/vinit/project_6/project_6.srcs/sim_1/new/tb_priority_decoder.v

```
1  `timescale 1ns / 1ps
2
3  module tb_priority_decoder(
4      );
5      reg a,b;
6      wire d0,d1,d2,d3;
7
8      priority_decoder uut(a,b,d0,d1,d2,d3);
9
10     initial
11     begin
12     a =0; b=0;
13     #10
14      a =0; b=1;
15     #10
16      a =1; b=0;
17     #10
18      a =1; b=1;
19     #10
20     $finish;
21     end
22 endmodule
23
```

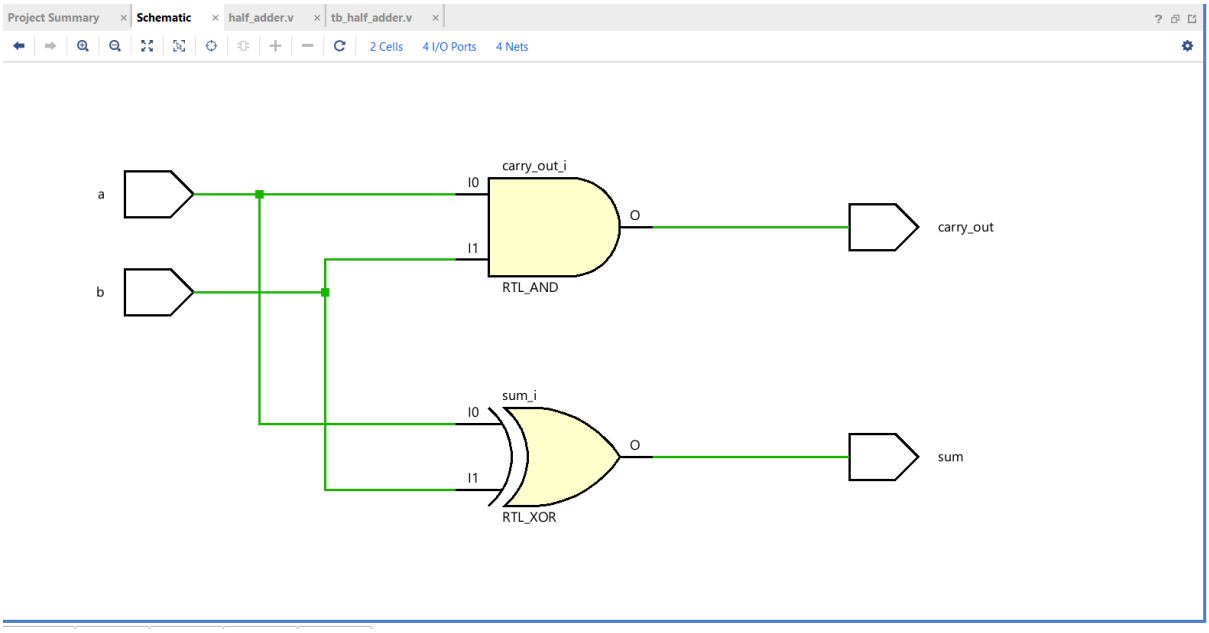


6A) HALF ADDER

Project Summary | half_adder.v

C:/Users/vinit/project_half_add/project_half_add.srcs/sources_1/new/half_adder.v

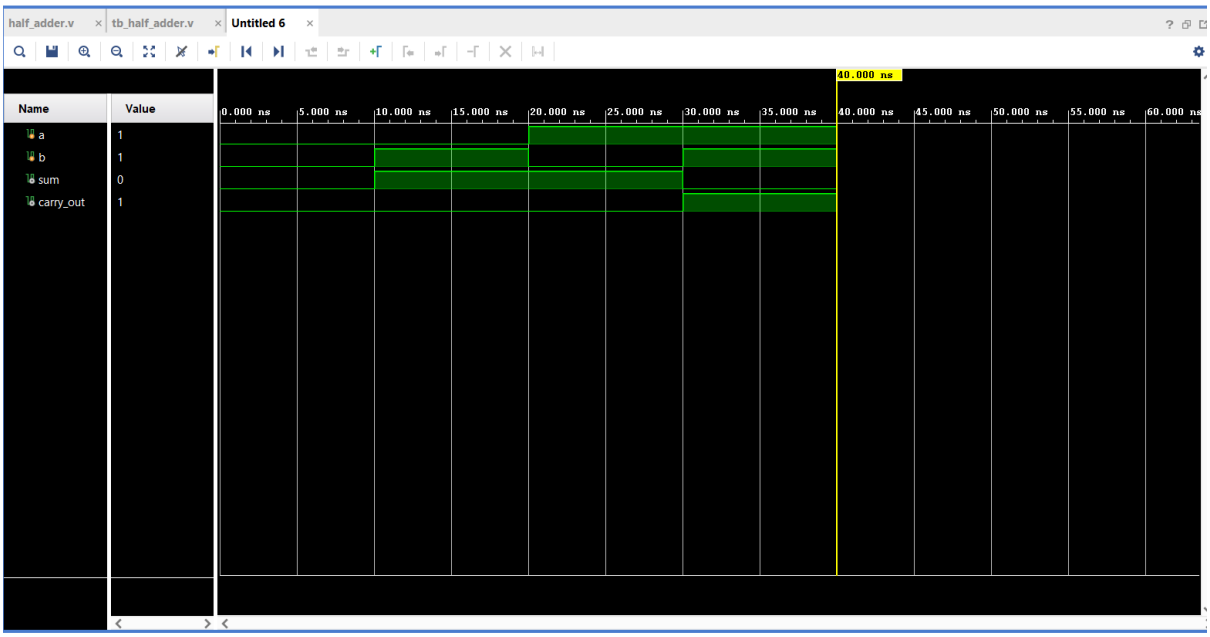
```
1  `timescale 1ns / 1ps
2
3
4  module half_adder(
5  input a,b,output sum,carry_out
6      );
7
8     assign sum =a^b;
9     assign carry_out=a&b;
10 endmodule
11
```



Project Summary | Schematic | half_adder.v | tb_half_adder.v *

C:/Users/vinit/project_half_add/project_half_add.srcs/sim_1/new/tb_half_adder.v

```
1  `timescale 1ns / 1ps
2
3  module tb_half_adder(
4      );
5      reg a,b;
6      wire a,b,sum,carry_out;
7      half_adder uut(a,b,sum,carry_out);
8
9
10
11     initial
12     begin
13     a=0;b=0;
14     #10
15     a=0;b=1;
16     #10
17     a=1;b=0;
18     #10
19     a=1;b=1;
20     #10
21     $finish;
22     end
23 endmodule
24
```

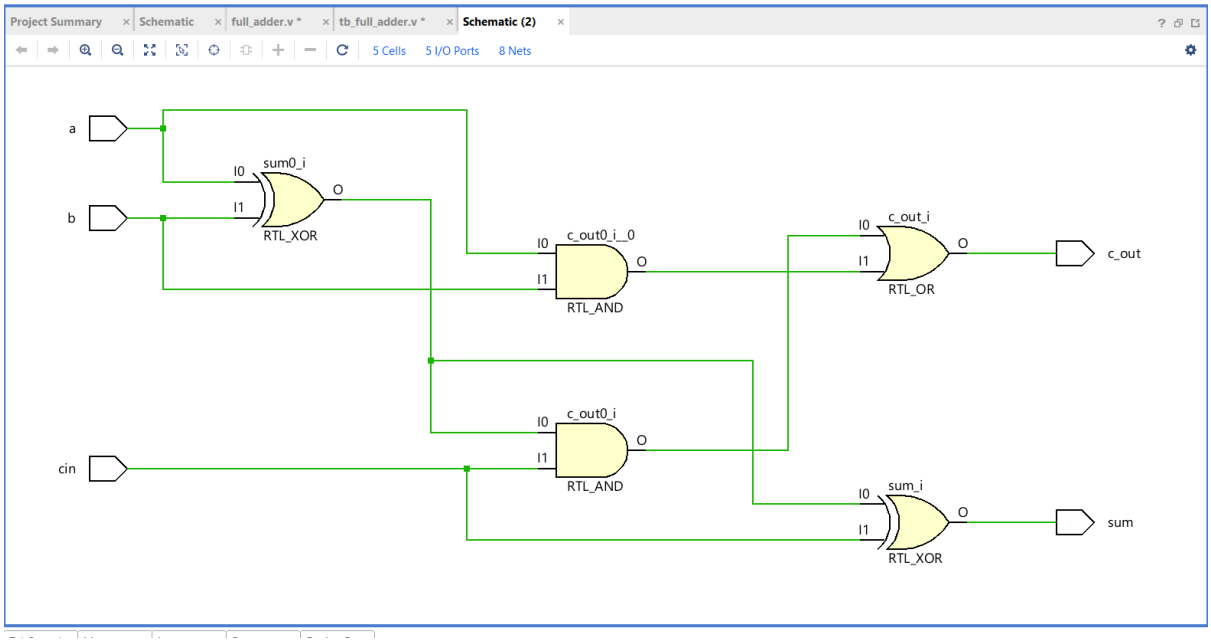


6B) FULL ADDER

full_adder.v * | tb_full_adder.v | Untitled 8

C:/Users/vinit/project_full_add/project_full_add.srcs/sources_1/new/full_adder.v

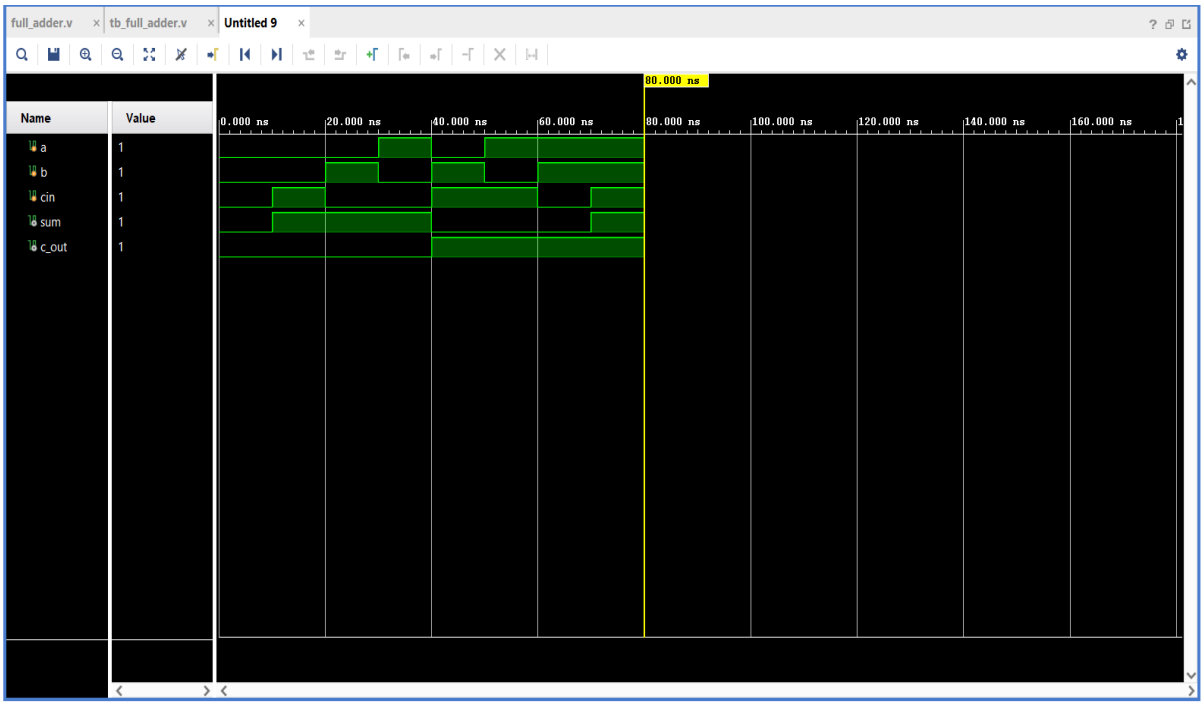
```
1   `timescale 1ns / 1ps
2
3   module full_adder(
4   input a,b,cin,output sum,c_out
5       );
6       assign sum =(a^b)^cin;
7       assign c_out =((a^b)&cin|a&b);
8   endmodule
9
```



Project Summary | full_adder.v | tb_full_adder.v *

C:/Users/vinit/project_full_add/project_full_add.srcs/sim_1/new/tb_full_adder.v

```
1  `timescale 1ns / 1ps
2
3  module tb_full_adder(
4
5      );
6      reg a,b,cin;
7      wire a,b,cin,sum,c_out;
8          full_adder uut(a,b,cin,sum,c_out);
9          initial
10     begin
11     a=0;b=0; cin=0;
12     #10
13      a=0;b=0; cin=1;
14     #10
15      a=0;b=1; cin=0;
16     #10
17      a=1;b=0; cin=0;
18     #10
19      a=0;b=1; cin=1;
20     #10
21      a=1;b=0; cin=1;
22     #10
23       a=1;b=1; cin=0;
24     #10
25      a=1;b=1; cin=1;
26     #10
27     $finish;
28     end
29           endmodule
30
```



7A) HALF SUBTRACTOR

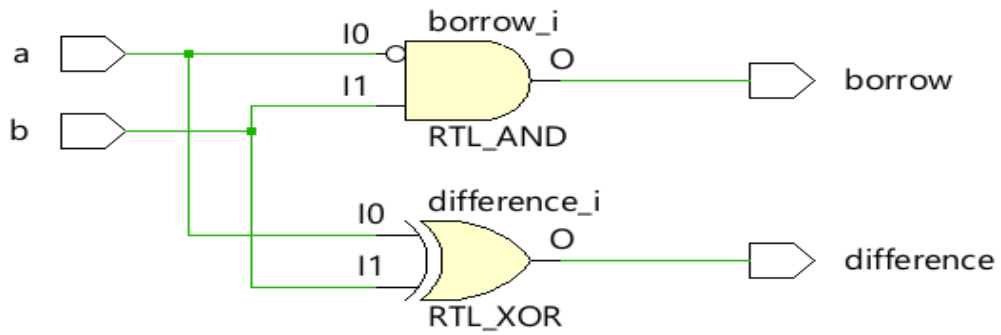
Project Summary | half_sub.v

C:/Users/vinit/project_half_sub/project_half_sub.srcs/sources_1/new/half_sub.v

```
1  `timescale 1ns / 1ps
2
3
4
5  module half_sub(
6  input a,b,output difference,borrow
7      );
8
9      assign difference = a^b;
10     assign borrow= (~a)&b;
11 endmodule
12
```

nmary | Schematic | half_sub.v | tb_half_sub.v

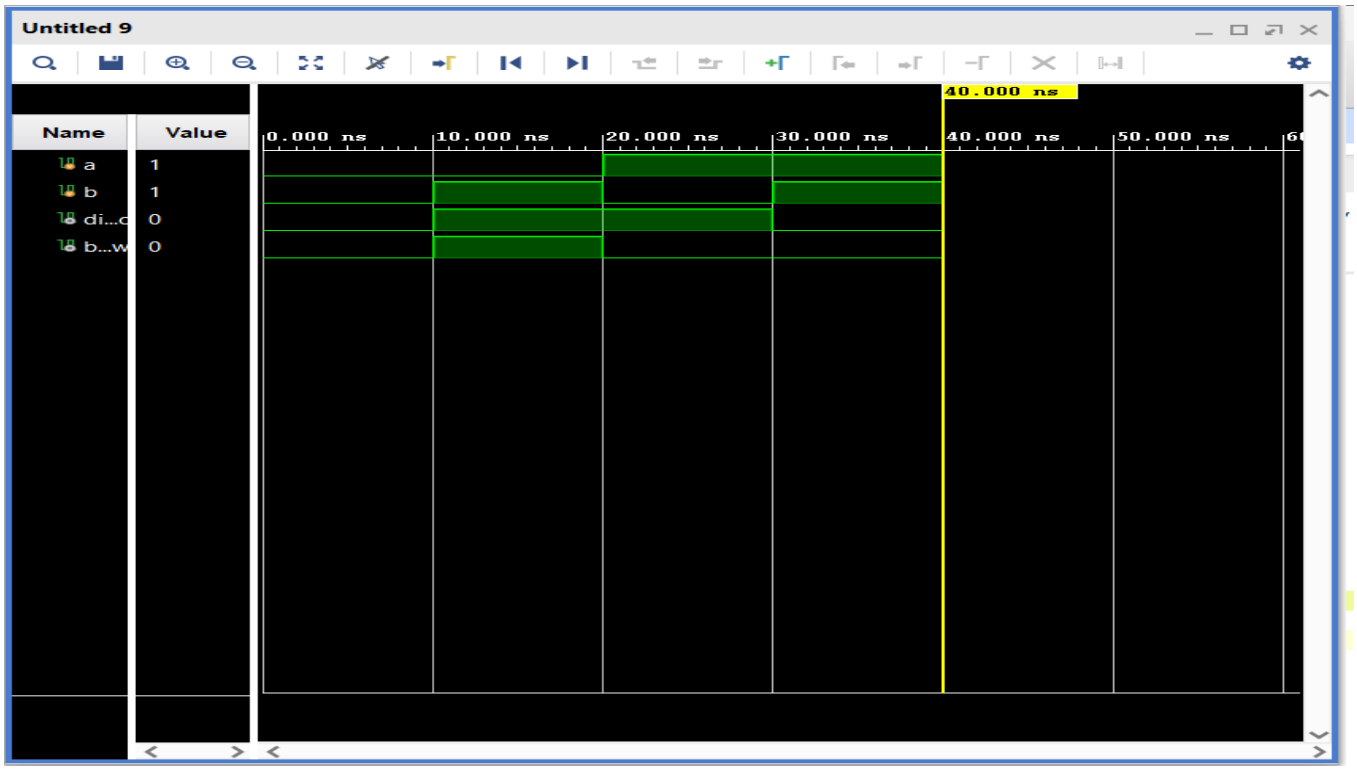
2 Cells | 4 I/O Ports



Project Summary | half_sub.v | tb_half_sub.v

sers/vinit/project_half_sub/project_half_sub.srcs/sim_1/new/tb_half_sub.v

```
1  `timescale 1ns / 1ps
2  module tb_half_sub(
3   );
4       reg a,b;
5       wire a,b,difference,borrow;
6       half_sub uut(a,b,difference,borrow);
7           initial
8       begin
9       a=0;b=0;
10      #10
11      a=0;b=1;
12      #10
13      a=1;b=0;
14      #10
15      a=1;b=1;
16      #10
17      $finish;
18      end
19 endmodule
20
```

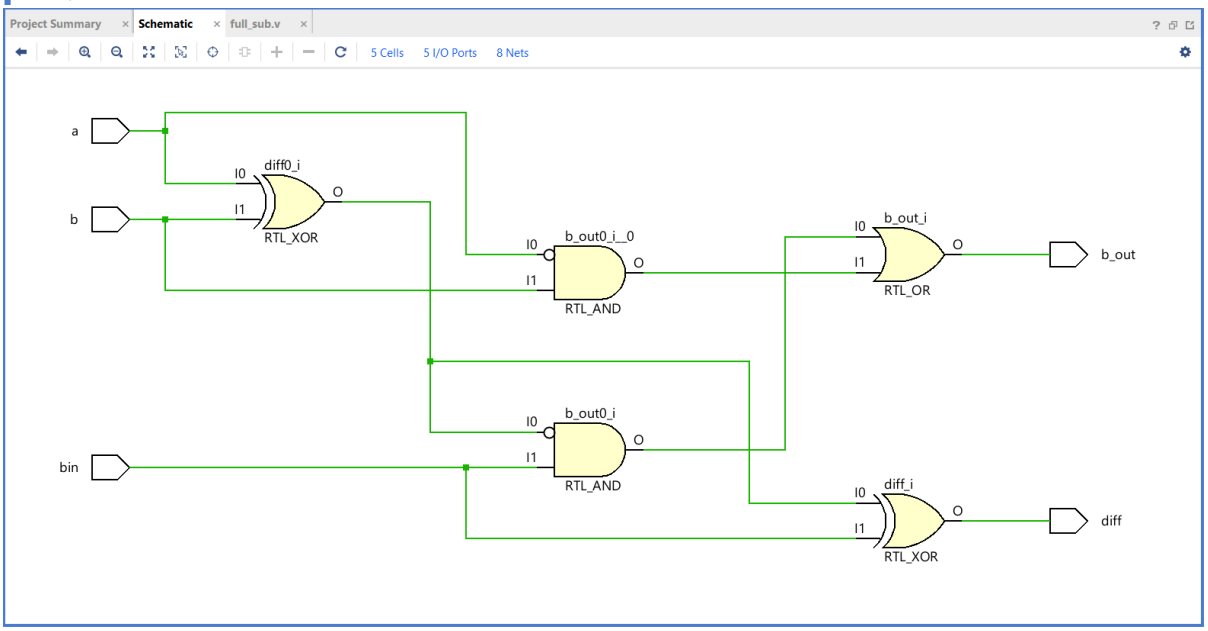


7B) FULL SUBTRACTOR

Project Summary | full_sub.v *

C:/Users/vinit/project_full_sub/project_full_sub.srcs/sources_1/new/full_sub.v

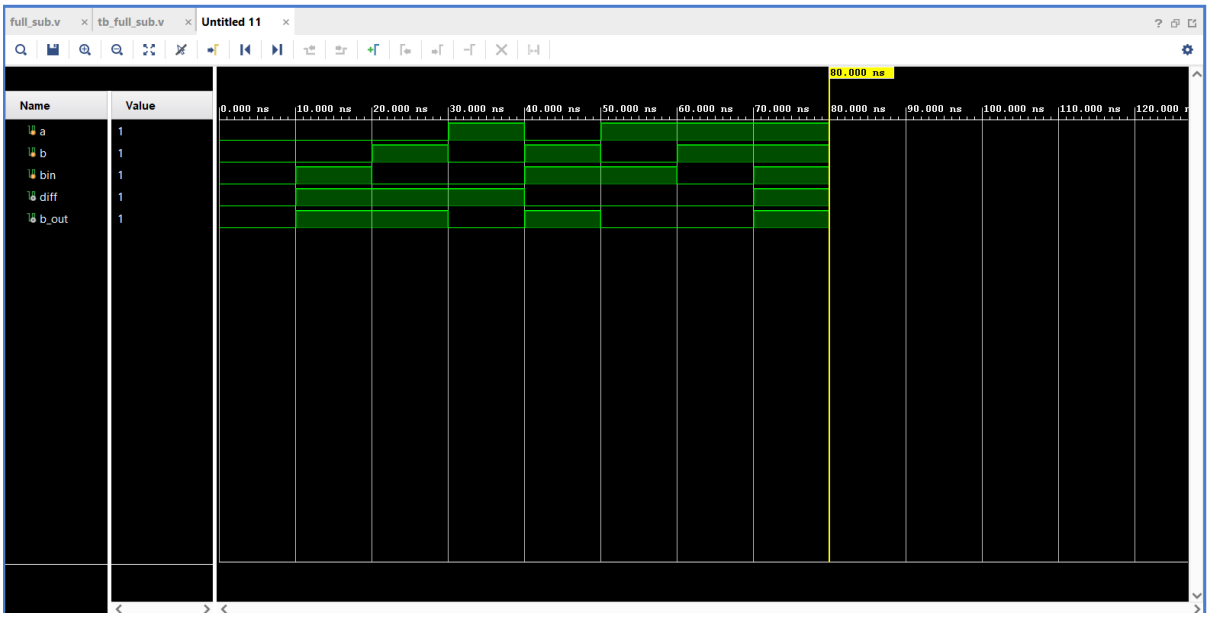
```
1  `timescale 1ns / 1ps
2
3  module full_sub(
4      input a,
5      input b,
6      input bin,
7      output diff,
8      output b_out
9      );
10
11     assign diff =(a^b)^bin;
12     assign b_out=(~(a^b)&bin)|(~a&b);
13 endmodule
14
```



Project Summary | Schematic | full_sub.v | tb_full_sub.v *

C:/Users/vinit/project_full_sub/project_full_sub.srcs/sim_1/new/tb_full_sub.v

```
1  `timescale 1ns / 1ps
2  module tb_full_sub(
3      );
4          reg a,b,bin;
5      wire a,b,bin,diff,b_out;
6
7      full_sub uut(a,b,bin,diff,b_out);
8
9      initial
10     begin
11     a=0;b=0; bin=0;     #10
12      a=0;b=0; bin=1;     #10
13      a=0;b=1; bin=0;     #10
14      a=1;b=0; bin=0;     #10
15      a=0;b=1; bin=1;     #10
16      a=1;b=0; bin=1;     #10
17       a=1;b=1; bin=0;     #10
18      a=1;b=1; bin=1;     #10
19     $finish;
20     end
21
22
23 endmodule
24
25
```

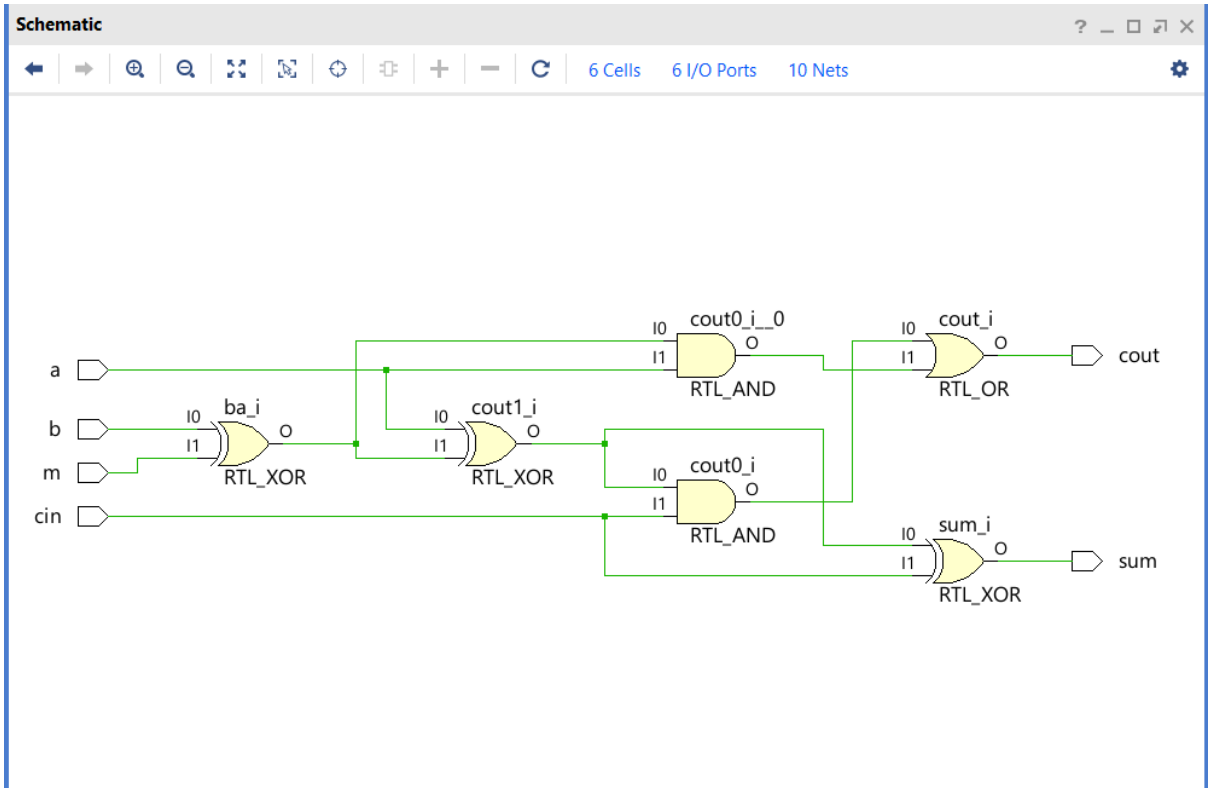


8A) UNIVERSAL ADDER AND SUBTRACTOR(1BIT)

Project Summary | universal.v

C:/Users/vinit/project_universal_add_sub/project_universal_add_sub.srcs/sources_1/ne

```
1  `timescale 1ns / 1ps
2
3  module universal(
4  input a,b,cin,m, output cout,sum
5      );
6      wire ba;
7      assign ba =(b^m);
8      assign sum = a ^(ba) ^ cin;
9   assign cout = ((a ^ (ba)) & cin) | ((ba) & a);
10 endmodule
11
```

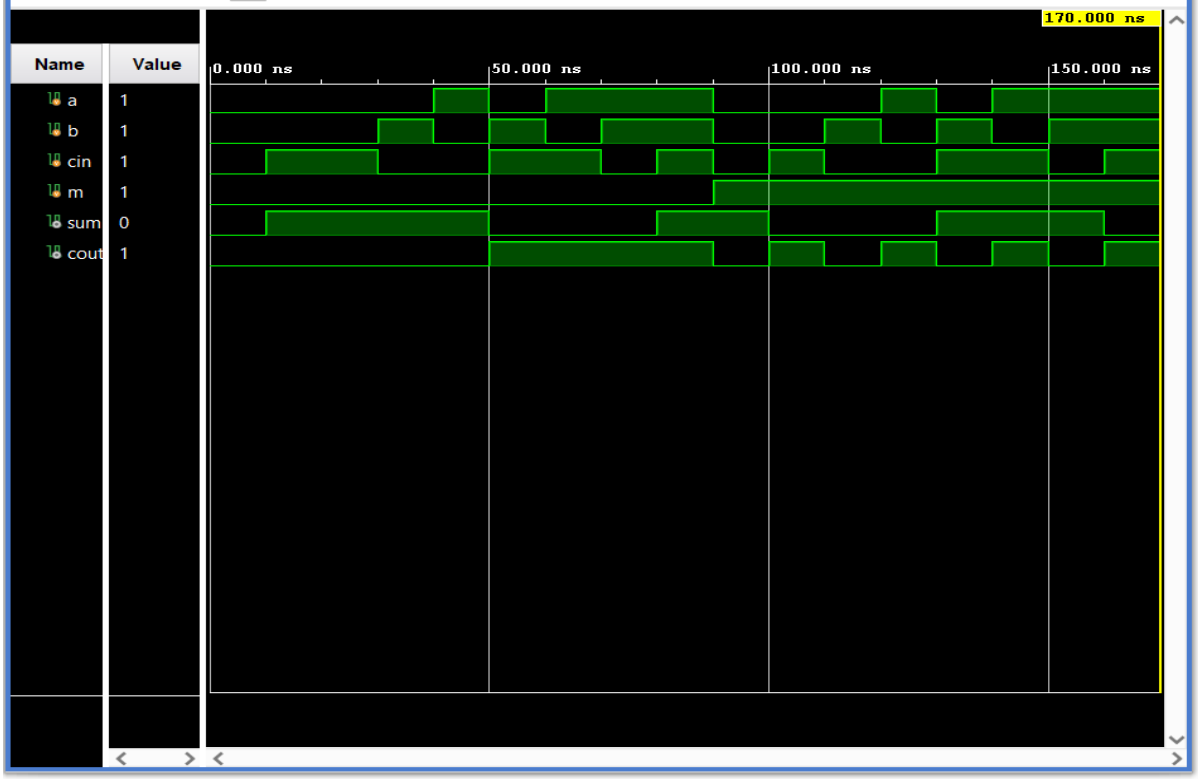


tb_universal.v

C:/Users/vinit/project_universal_add_sub/project_universal_add_sub.srcs/sim_1/new/tb_universal.v

```
1  `timescale 1ns / 1ps
2  module tb_universal(
3      );
4      reg a,b,cin,m;
5      wire sum,cout;
6      universal uut (a,b,cin,m,cout,sum);
7      initial
8      begin
9      a=0; b=0;cin=0;m=0;
10
11    #10      a=0; b=0;cin=1;m=0;
12     #10
13     #10      a=0; b=1;cin=0;m=0;
14    #10      a=1; b=0;cin=0;m=0;
15     #10      a=0; b=1;cin=1;m=0;
16    #10      a=1; b=0;cin=1;m=0;
17     #10      a=1; b=1;cin=0;m=0;
18     #10      a=1; b=1;cin=1;m=0;
19    #10
20     // subtractor
21     a=0; b=0;cin=0;m=1;
22     #10      a=0; b=0;cin=1;m=1;
23     #10      a=0; b=1;cin=0;m=1;
24     #10      a=1; b=0;cin=0;m=1;
25     #10       a=0; b=1;cin=1;m=1;
26     #10      a=1; b=0;cin=1;m=1;
27     #10 a=1; b=1;cin=0;m=1;
28     #10 a=1; b=1;cin=1;m=1;
29     #10
30     $finish;
31    end
```

Untitled 11

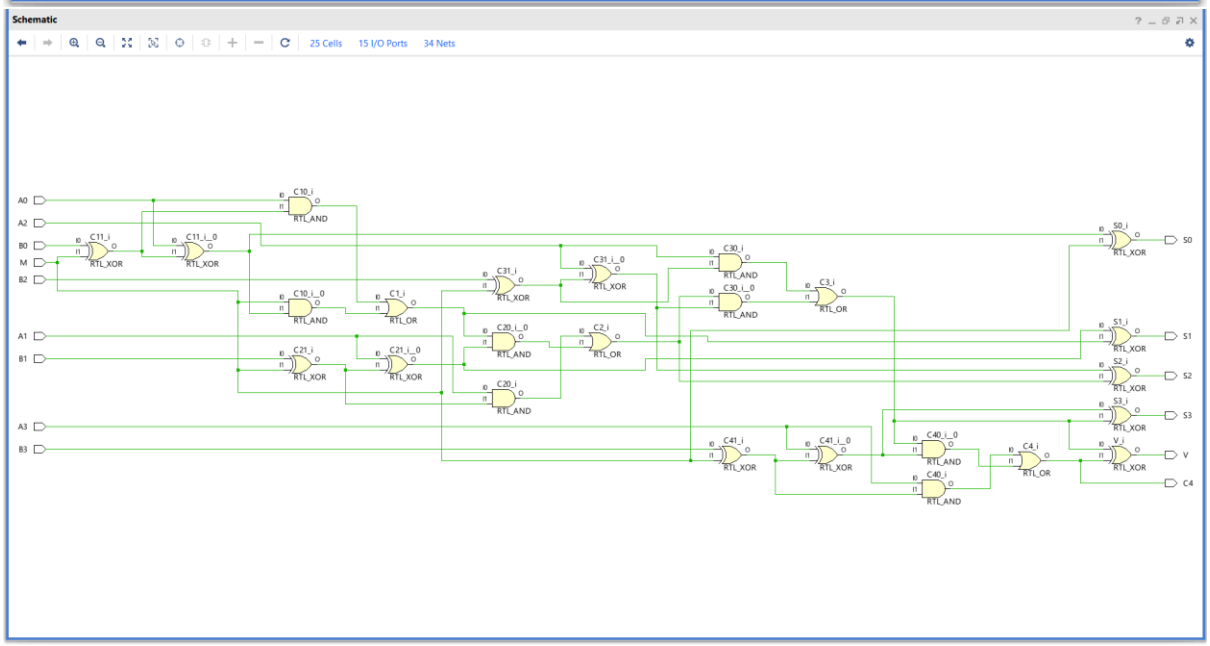


8 B) UNIVERSAL ADDER AND SUBTRACTOR (4BIT)

universal4bit.v

C:/Users/vinit/project_4bituniversal/project_4bituniversal.srcs/sources_1/new/universal4bit.v

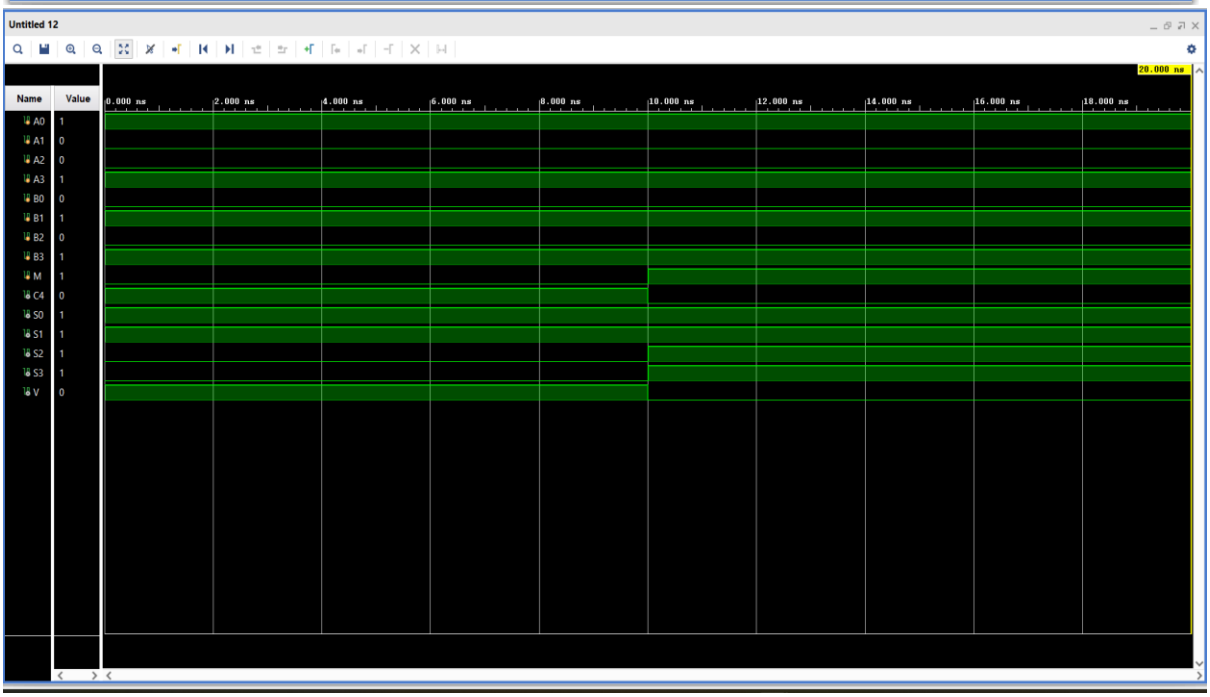
```
1  `timescale 1ns / 1ps
2
3  module universal4bit(
4  input A0, A1, A2, A3, B0, B1, B2, B3, M,
5  output C4, S0, S1, S2, S3,V
6      );
7      assign C0= M;
8      assign S0= A0 ^ (B0 ^ M) ^ C0;
9      assign C1= (A0 & (B0 ^ M)) | (C0 & (A0 ^ (B0 ^ M)));
10
11     assign S1= A1 ^ (B1 ^ M) ^ C1;
12     assign C2= (A1 & (B1 ^ M)) | (C1 & (A1 ^ (B1 ^ M)));
13
14     assign S2= A2 ^ (B2 ^ M) ^ C2;
15     assign C3= (A2 & (B2 ^ M)) | (C2 & (A2 ^ (B2 ^ M)));
16
17     assign S3= A3 ^ (B3 ^ M) ^ C3;
18     assign C4= (A3 & (B3 ^ M)) | (C3 & (A3 ^ (B3 ^ M)));
19
20     assign V= C3 ^ C4;
21
22
23
24 endmodule
25
```



tb_universal4bit.v

C:/Users/vinit/project_4bituniversal/project_4bituniversal.srcs/sim_1/new/tb_universal4bit.v

```
1  `timescale 1ns / 1ps
2
3  module tb_universal4bit(
4
5      );
6
7  reg A0, A1, A2, A3, B0, B1, B2, B3, M;
8  wire C4, S0, S1, S2, S3,V;
9
10 universal4bit uut(A0, A1, A2, A3, B0, B1, B2, B3, M,C4, S0, S1, S2, S3,V);
11
12 initial begin
13 // ADD: 9 and 10 [1001 + 1010]
14 A3=1; A2=0; A1=0; A0=1;
15 B3=1; B2=0; B1=1; B0=0;
16 M=0;
17 #10
18
19 // SUBTRACT: 10 from 9 [1001- 1010]
20 A3=1; A2=0; A1=0; A0=1;
21 B3=1; B2=0; B1=1; B0=0;
22 M=1;
23 #10
24 $finish;
25
26 end
27
28 endmodule
29
```

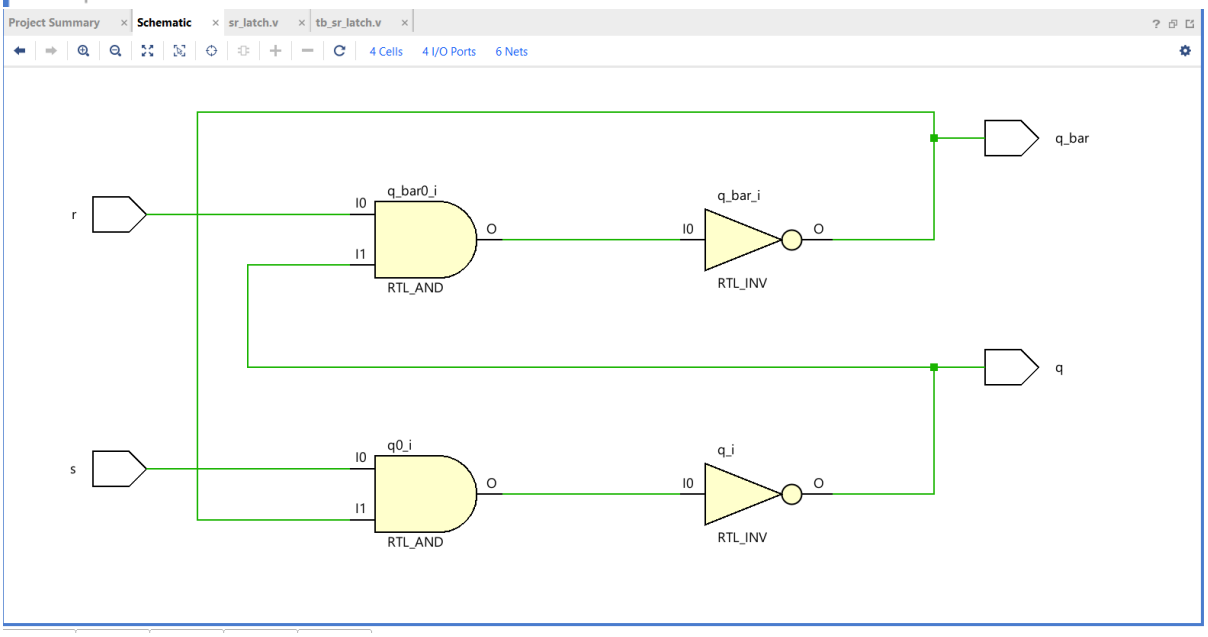


9) SR LATCH

Project Summary | sr_latch.v * | tb_sr_latch.v *

C:/Users/vinit/project_sr_latch/project_sr_latch.srcs/sources_1/new/sr_latch.v

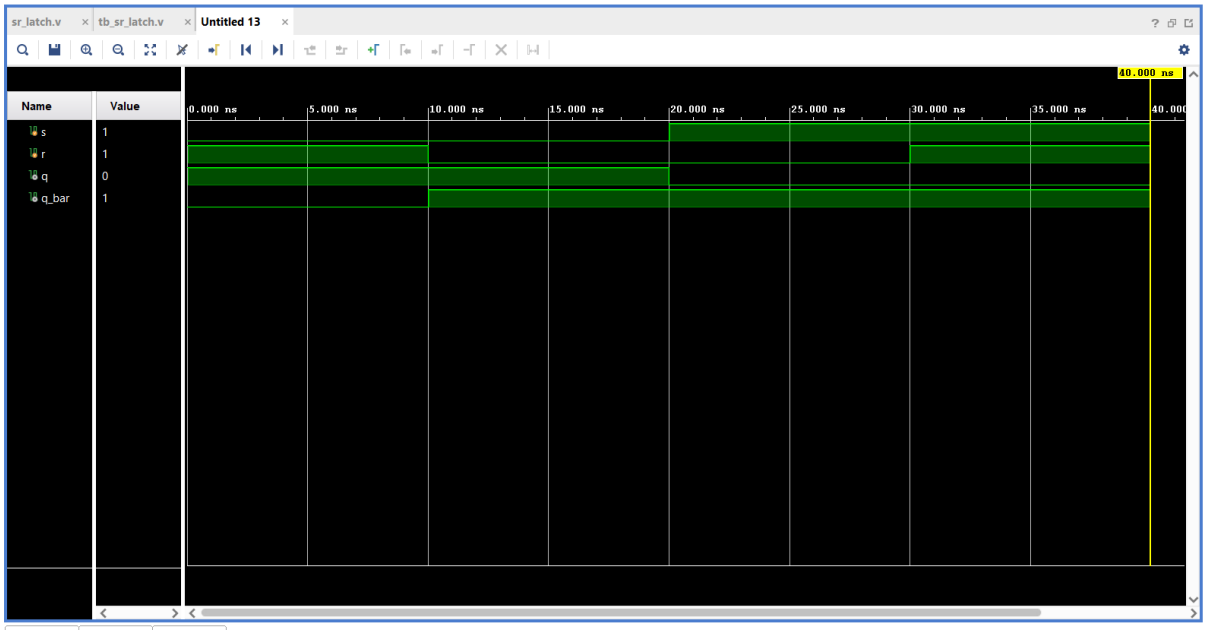
```
1  `timescale 1ns / 1ps
2
3
4  module sr_latch(
5  input s,r,output q,q_bar
6      );
7      assign q=~(s&q_bar);
8      assign q_bar=~(r&q);
9  endmodule
10
```



Project Summary | sr_latch.v * | tb_sr_latch.v *

C:/Users/vinit/project_sr_latch/project_sr_latch.srcs/sim_1/new/tb_sr_latch.v

```
1  `timescale 1ns / 1ps
2
3  module tb_sr_latch(
4
5      );
6      reg s,r;
7      wire s,r,q,q_bar;
8
9      sr_latch uut(s,r,q,q_bar);
10
11     initial
12     begin
13             s=0;r=1;
14     #10
15         s=0;r=0;
16         #10
17     s=1;r=0;
18     #10
19     s=1;r=1;
20     #10
21     $finish;
22     end
23 endmodule
24
```

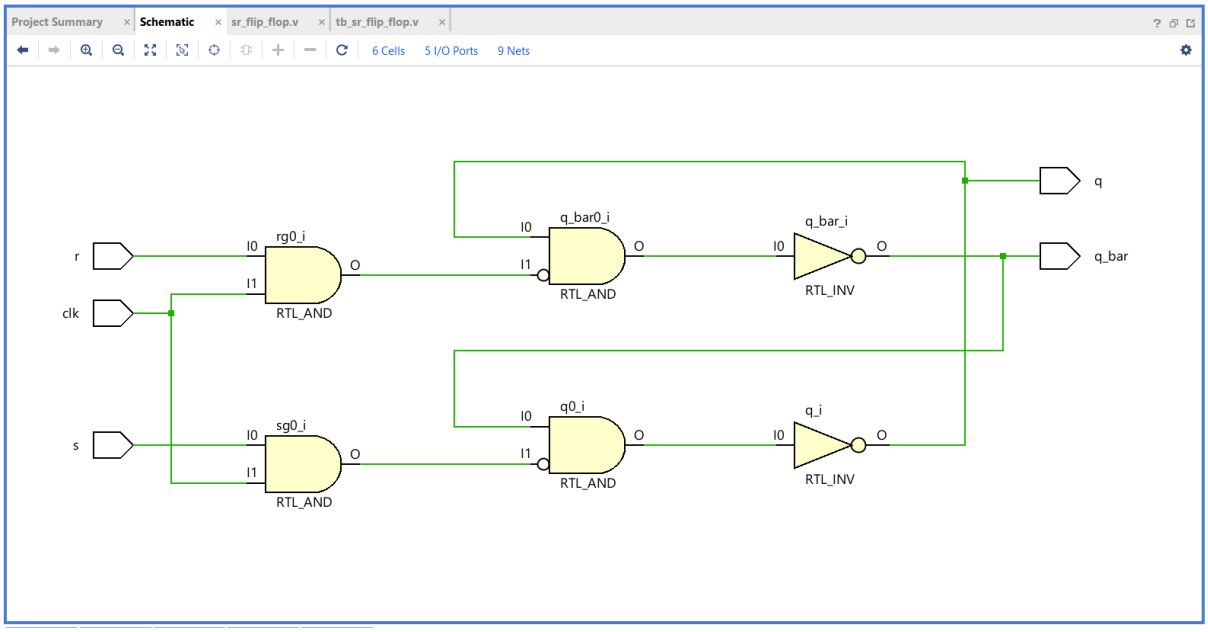


10A) S R FLIP FLOP

Project Summary | sr_flip_flop.v * | tb_sr_flip_flop.v

C:/Users/vinit/project_flip_flop/project_flip_flop.srcs/sources_1/new/sr_flip_flc

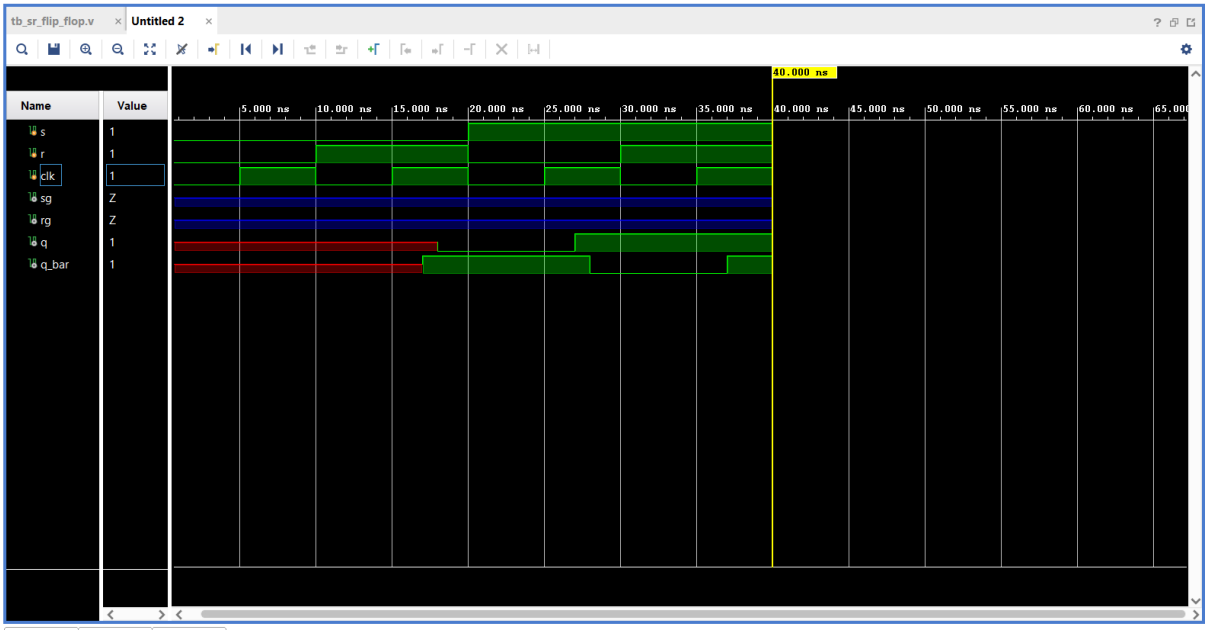
```
1  `timescale 1ns / 1ps
2
3  module sr_flip_flop(
4      input s,
5      input r,
6      input clk,
7      output q,
8      output q_bar
9      );
10     wire sg,rg;
11     assign #1 sg= ~(s&clk);
12     assign #1 rg=~(r&clk);
13     assign #1 q=~(q_bar&sg);
14     assign #1 q_bar=~(q&rg);
15 endmodule
16
```



Project Summary | Schematic | sr_flip_flop.v | tb_sr_flip_flop.v

C:/Users/vinit/project_flip_flop/project_flip_flop.srcs/sim_1/new/tb_sr_flip_flop.v

```
1  `timescale 1ns / 1ps
2  module tb_sr_flip_flop(
3
4      );
5      reg s,r,clk;
6      wire s,r,clk,sg,rg,q,q_bar;
7
8      sr_flip_flop uut(s,r,clk,q,q_bar);
9
10     initial
11     begin
12     clk= 0;
13     forever #5 clk=~clk;
14     end
15
16     initial
17     begin
18      s=0;r=0;
19     #10
20     s=0;r=1;
21     #10
22
23     s=1;r=0;
24     #10
25     s=1;r=1;
26     #10
27     $finish;
28     end
29
30 endmodule
```



10 B) J K FLIP FLOP

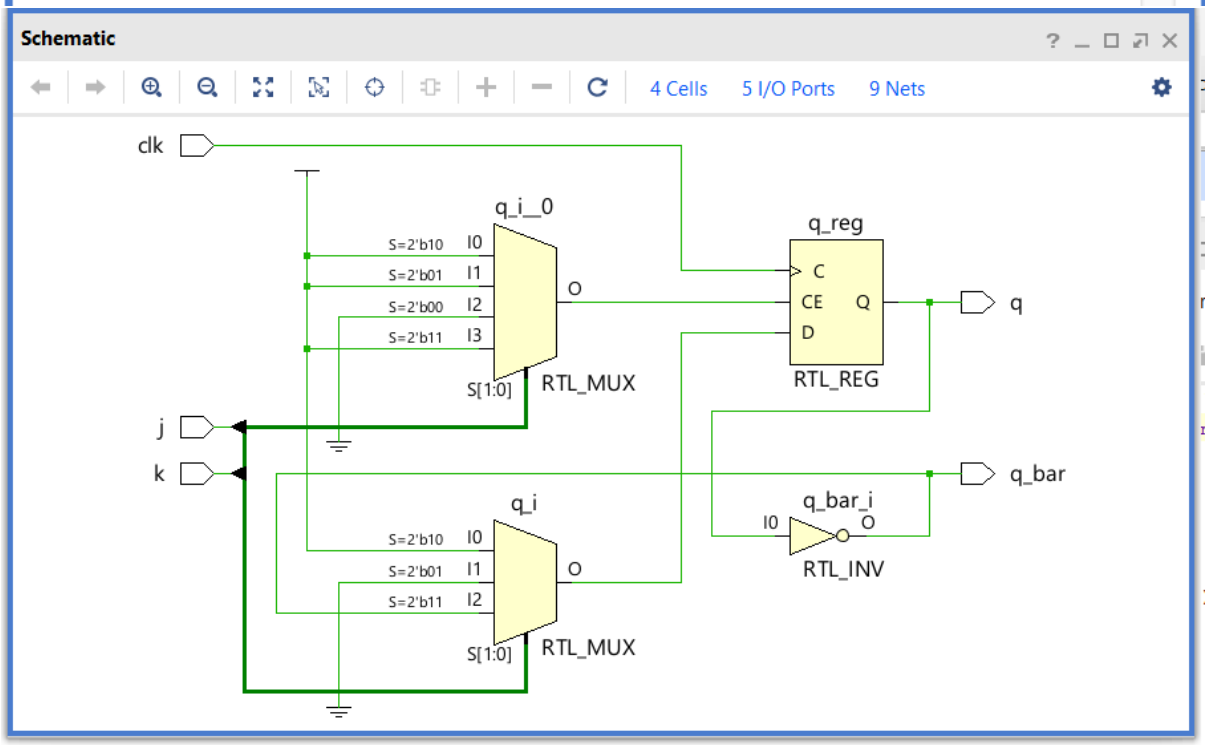
Project Summary | j_k.v

C:/Users/vinit/project_j_k_flip/project_j_k_flip.srcs/sources_1/new/j_k.v

```
1  `timescale 1ns / 1ps
2  module j_k(
3      input j,
4      input k,
5      input clk,
6      output reg q,
7      output q_bar
8  );
9          assign q_bar=~q;
10         always @(posedge clk) begin
11         case({j,k})
12         2'b10: q<=1'b1;
13         2'b01: q<=1'b0;
14         2'b00: q<=q;
15         2'b11: q<=q_bar;
16         endcase
17          end
18 endmodule
19
```

Schematic

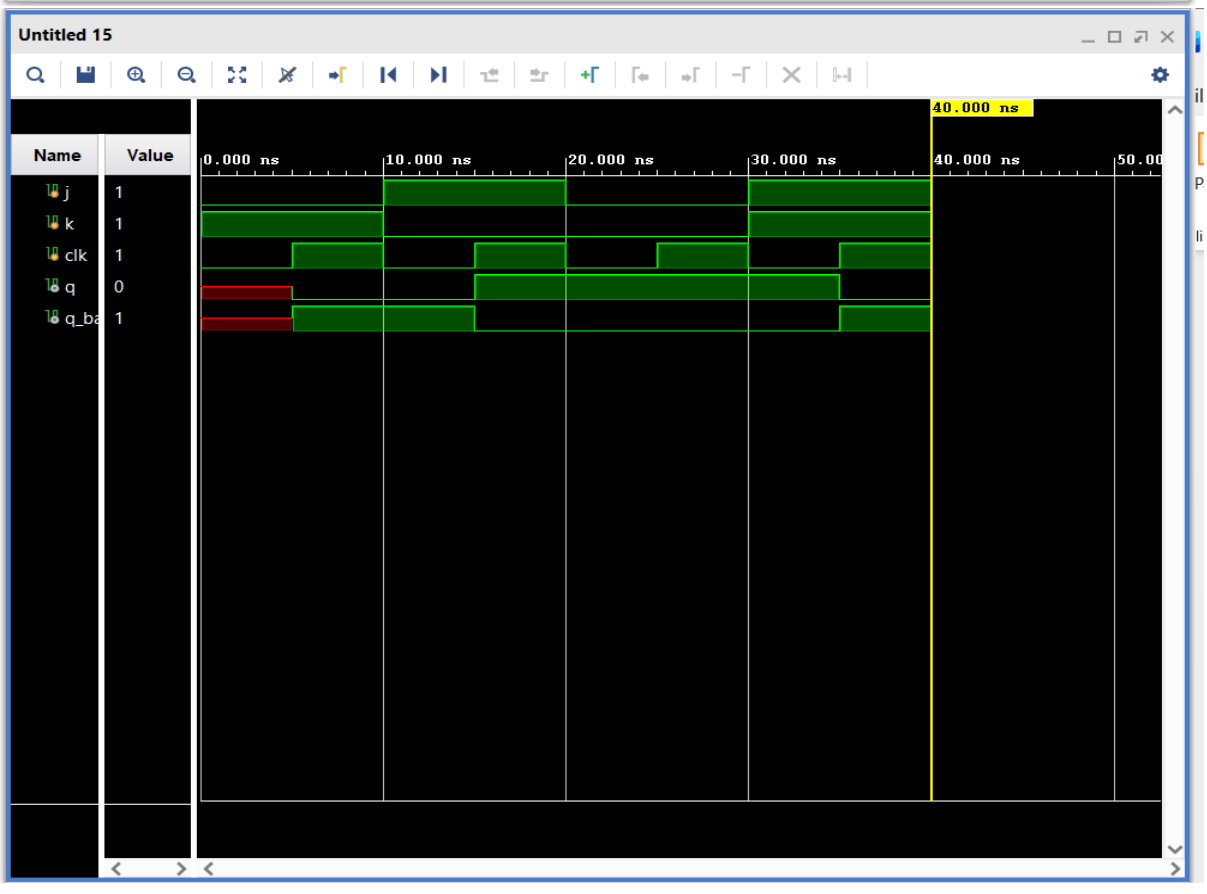
4 Cells | 5 I/O Ports | 9 Nets



tb_j_k.v

C:/Users/vinit/project_j_k_flip/project_j_k_flip.srcs/sim_1/new/tb_j_k.v

```
1  `timescale 1ns / 1ps
2  module tb_j_k();
3      reg j,k,clk;
4      wire q,q_bar;
5          j_k uut(.j(j), .k(k), .clk(clk), .q(q), .q_bar(q_bar));
6          initial
7      begin
8          clk = 0;
9          forever #5 clk = ~clk;
10     end
11     initial begin
12
13         j=0; k=1;
14
15         #10;
16         j=1; k=0;
17
18         #10;
19         j=0; k=0;
20
21         #10;
22         j=1; k=1;
23
24         #10;
25         $finish;
26     end
27 endmodule
```



11 A) D FLIP FLOP

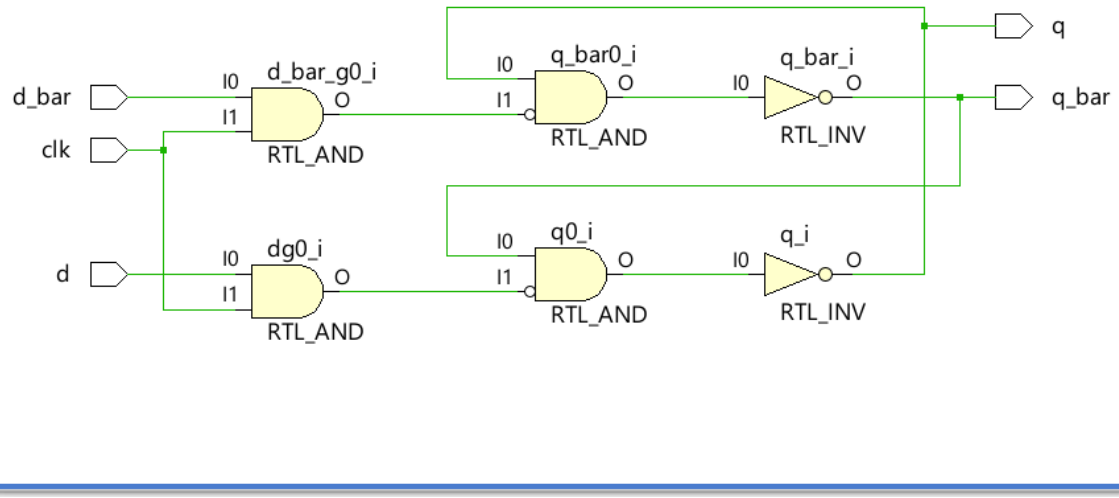
d_flip_gap.v

C:/Users/vinit/project_d_flip_flop/project_d_flip_flop.srcs/sources_1/new/d_flip_gap.v

```
1  `timescale 1ns / 1ps
2
3  module d_flip_gap(
4  input d,d_bar,clk,output q,q_bar
5      );
6      wire dg,d_bar_g;
7      assign #1 dg=~(d&clk);
8      assign  #1 d_bar_g=~(d_bar&clk);
9      assign  #1 q= ~(q_bar&dg);
10     assign  #1 q_bar=~(q&d_bar_g);
11 endmodule
```

Schematic

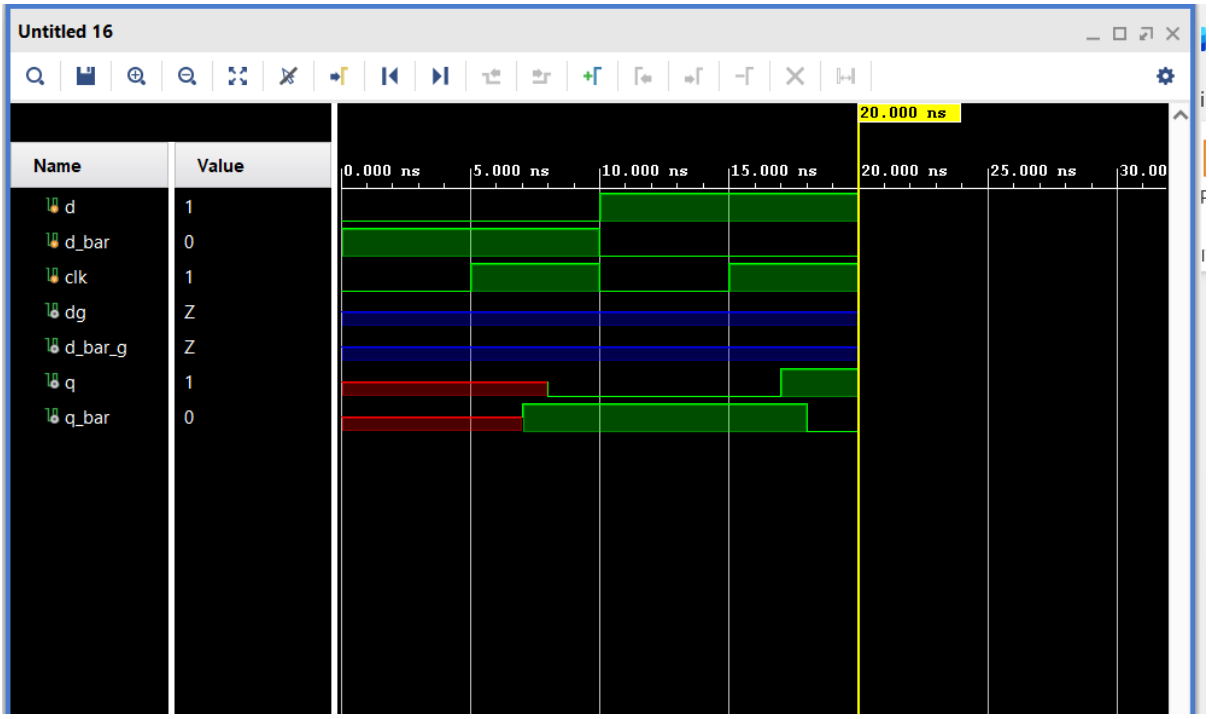
6 Cells 5 I/O Ports 9 Nets



Project Summary | tb_d_flip_gap.v

C:/Users/vinit/project_d_flip_flop/project_d_flip_flop.srcs/sim_1/new/tb_d_flip_gap.v

```
1  `timescale 1ns / 1ps
2
3  module tb_d_flip_gap(
4
5      );
6      reg d,d_bar,clk;
7      wire dg,d_bar_g,q,q_bar;
8
9      d_flip_gap uut(d,d_bar,clk,q,q_bar);
10
11     initial
12     begin
13     clk=0;
14     forever
15     #5 clk =~clk;
16     end
17
18     initial
19     begin
20     d=0;d_bar=~d;
21     #10
22     d=1;d_bar=~d;
23     #10
24     $finish;
25      end
26 endmodule
27
28
```



b)

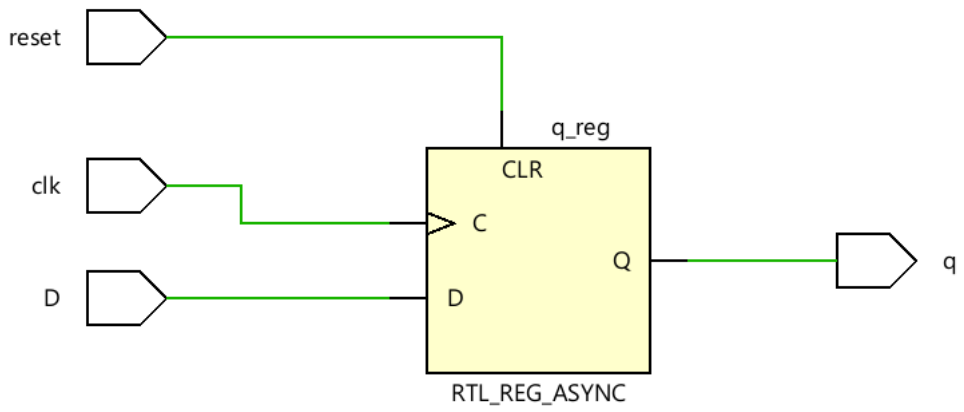
D_FLIP.v

C:/Users/vinit/project_D_FLIP_FLOP__/project_D_FLIP_FLOP__.srcs/sources_1/new/D_FLIP.v

```
1  `timescale 1ns / 1ps
2
3  module D_FLIP(
4  input D,clk,reset , output reg q
5      );
6      always @(posedge clk or posedge reset)
7      begin
8      if(reset)
9      q<=1'b0;
10     else
11     q<=D;
12     end
13 endmodule
14
```

Schematic

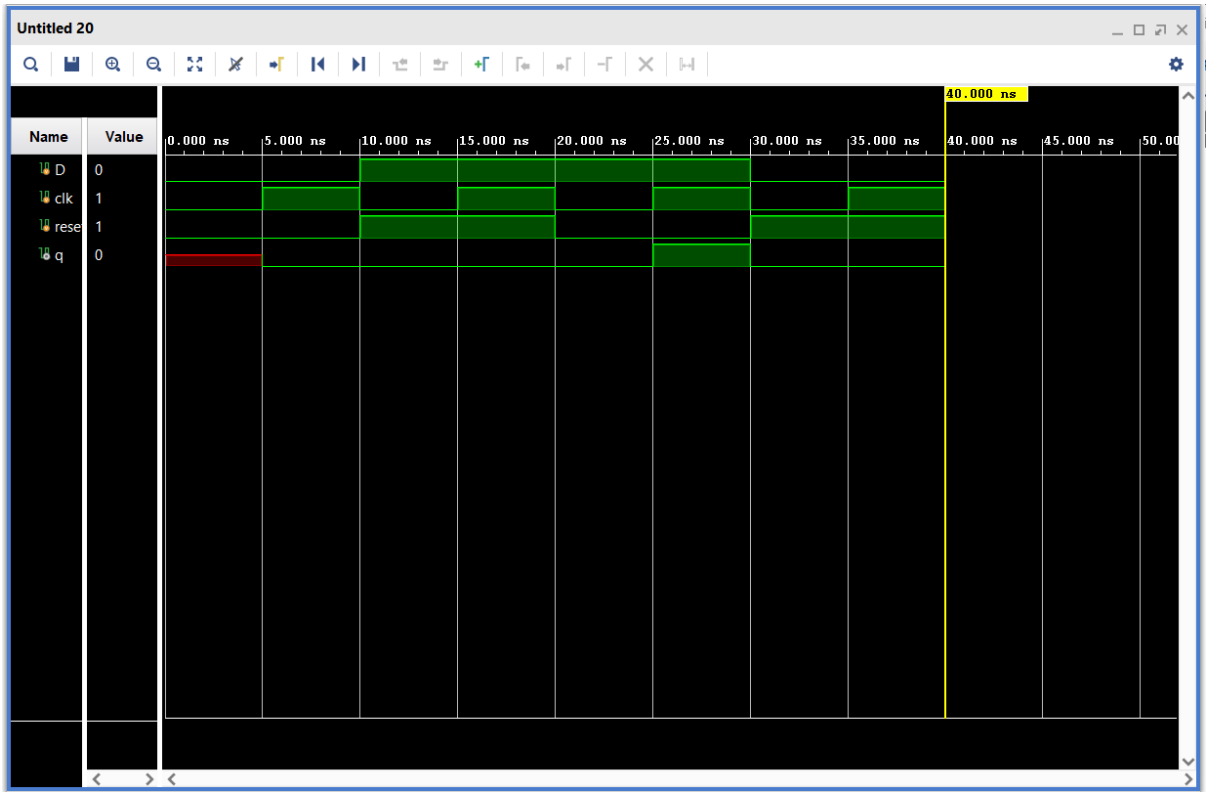
1 Cell | 4 I/O Ports | 4 Nets



Project Summary | tb_D_flip.v

/Users/vinit/project_D_FLIP_FLOP__/project_D_FLIP_FLOP__.srcs/sim_1/new/tb_D_flip.v

```
1  `timescale 1ns / 1ps
2
3  module tb_D_flip(
4       );
5       reg D,clk, reset;
6      wire q;
7
8      D_flip uut(D,clk,reset,q);
9
10     initial
11     begin
12     clk=0;
13     forever #5
14     clk=~clk;
15     end
16
17     initial
18     begin
19     D=0; reset =0;
20     #10
21     D=1;reset=1;
22     #10
23      D=1; reset =0;
24     #10
25     D=0;reset=1;
26     #10
27     $finish;
28     end
29 endmodule
30
```



11 B) T FLIP FLOP

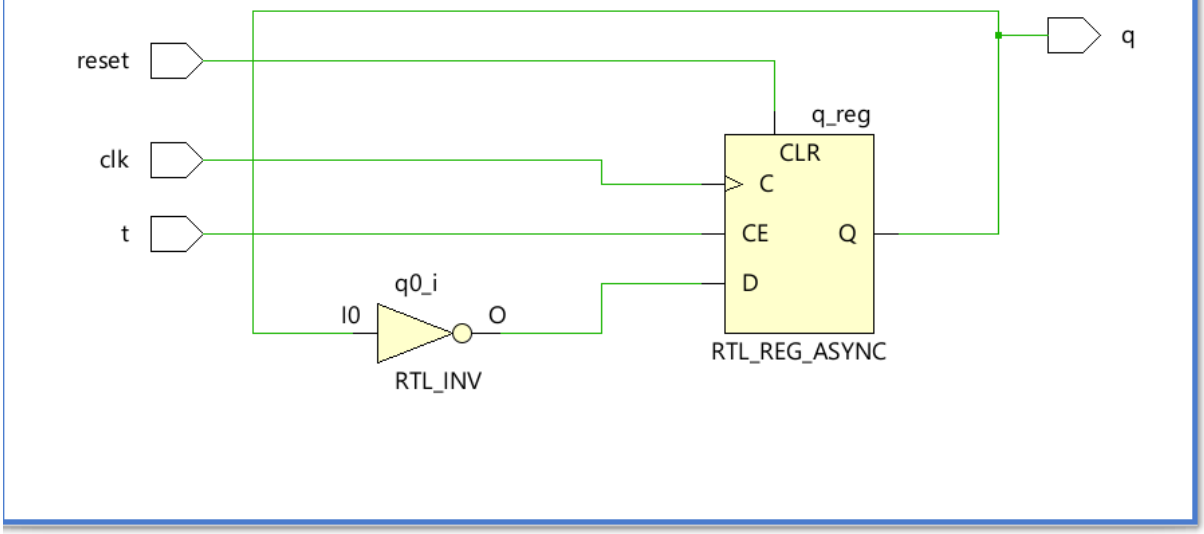
Project Summary | t_flip_flop.v

C:/Users/vinit/project_T/project_T.srcs/sources_1/new/t_flip_flop.v

```
1  `timescale 1ns / 1ps
2
3  module t_flip_flop(
4  input t,
5  input clk,
6  input reset,
7  output reg q
8      );
9      always @(posedge clk or posedge reset) begin
10     if(reset)//always at posivitive edge
11     q<=1'b0;
12     else if(t)
13     q<=~q;
14     end
15
16 endmodule
17
```

Schematic

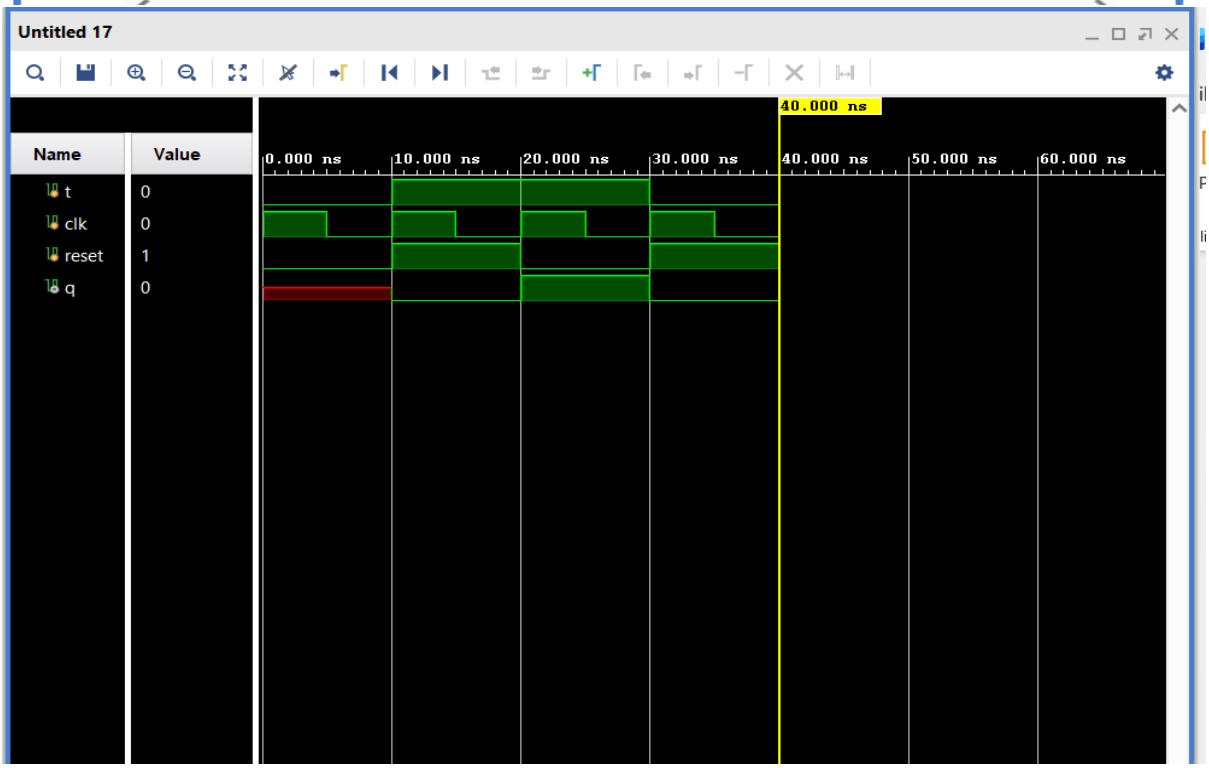
2 Cells | 4 I/O Ports | 5 Nets



Project Summary | t_flip_flop.v | tb_t_flip.v *

C:/Users/vinit/project_T/project_T.srcs/sim_1/new/tb_t_flip.v

```
1  `timescale 1ns / 1ps
2
3  module tb_t_flip(
4      );
5      reg t,clk,reset;
6      wire q;
7
8      t_flip_flop uut(t,clk,reset,q);
9
10     initial
11     begin
12     clk=1;
13         forever
14     #5 clk =~clk;
15     end
16     initial
17     begin
18     t=0;reset=0;
19     #10
20
21     t=1;reset=1;
22     #10
23     t=1;reset=0;
24     #10
25     t=0;reset=1;
26      #10
27      $finish;
28      end
29 endmodule
30
```



12) COUNTER DESIGN

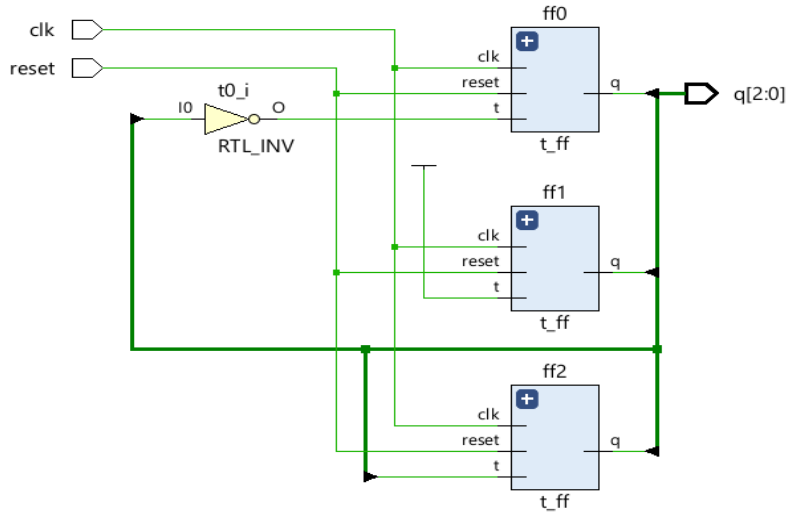
counter.v

C:/Users/vinit/project_counter/project_counter.srcs/sources_1/new/counter.v

```
1  `timescale 1ns / 1ps
2  module t_ff(
3  input t,
4  input clk,
5  input reset,
6  output reg q
7      );
8      always @(posedge clk or posedge reset) begin
9      if(reset)//always at posivitive edge
10     q<=1'b0;
11     else if(t)
12     q<=~q;
13     end
14
15 endmodule
16
17 module counter(
18 input clk, reset, output [2:0]q
19     );
20     wire t0,t1,t2;
21     assign t0=~q[1];
22     assign t1=1;
23     assign t2=q[1];
24
25      t_ff ff0(t0,clk,reset,q[0]);
26     t_ff ff1(t1,clk,reset,q[1]);
27      t_ff ff2(t2,clk,reset,q[2]);
28
29 endmodule
30
```

Schematic

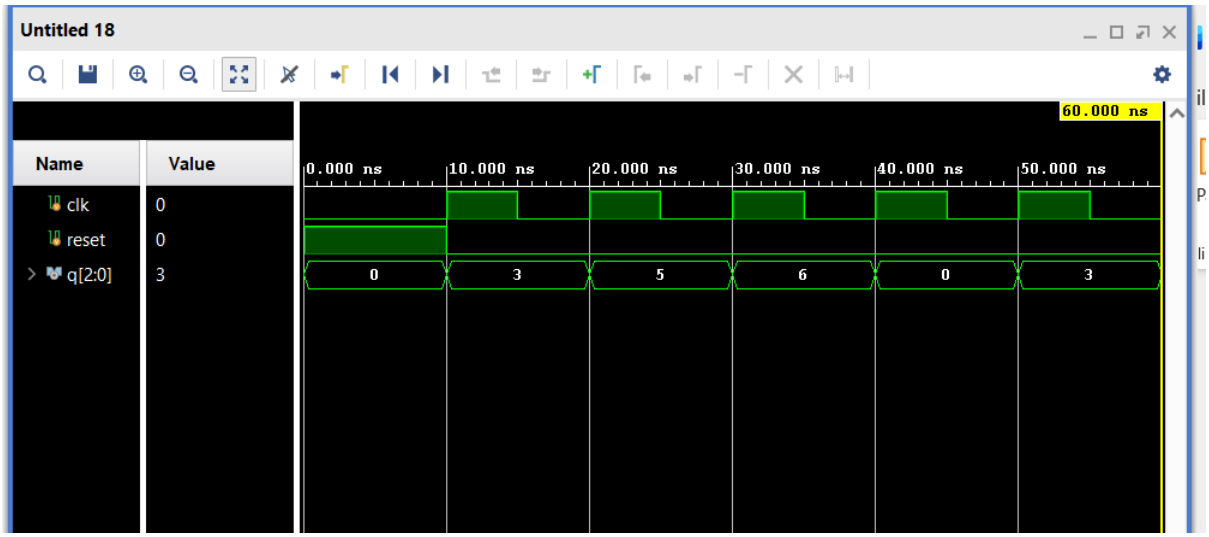
4 Cells 5 I/O Ports 7 Nets



Project Summary | tb_counter.v

C:/Users/vinit/project_counter/project_counter.srcs/sim_1/new/tb_counter.v

```
1  `timescale 1ns / 1ps
2  module tb_counter(
3      );
4
5      reg clk,reset;
6      wire [2:0]q;
7
8      counter uut(clk,reset,q);
9      initial
10     begin
11     clk=0;
12     #5
13     forever #5 clk=~clk;
14     end
15
16     initial
17     begin
18     reset =1;
19     #10
20     reset=0;
21     #50
22     $finish;
23     end
24
25 endmodule
26
```

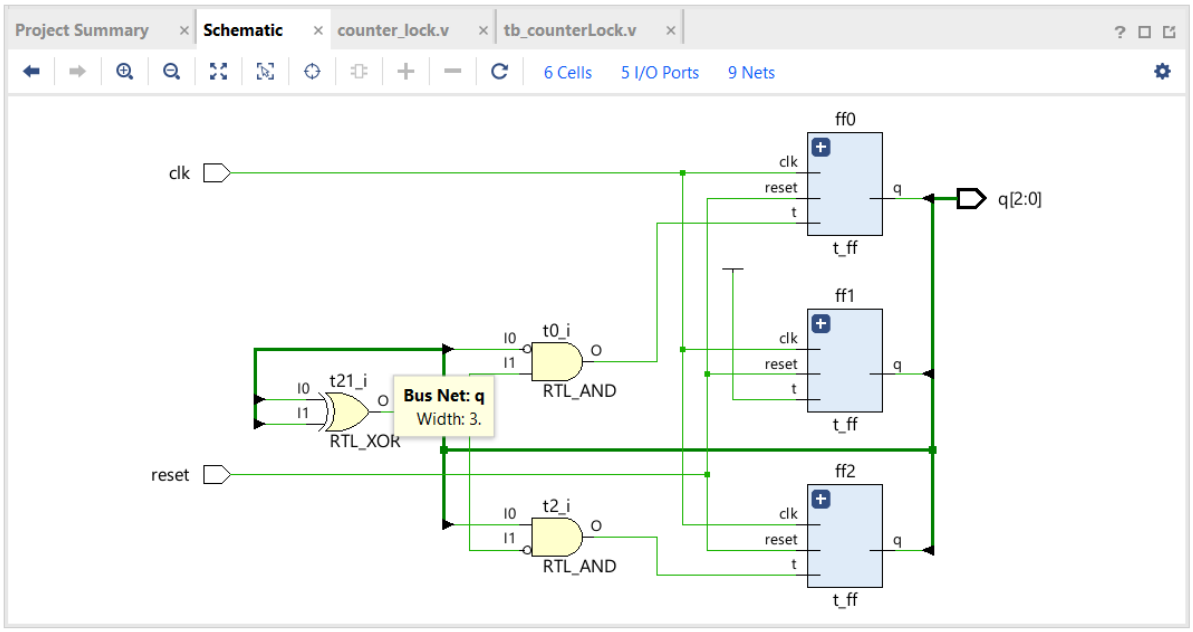


12 B) LOCKOUT OF THIS COUNTER(if is it going)

counter_lock.v | tb_counterLock.v | Untitled 2

C:/Users/vinit/project_counterlockout/project_counterlockout.srcs/sources_1/new/counter_lock.v

```
1   `timescale 1ns / 1ps
2   module t_ff(
3   input t,
4   input clk,
5   input reset,
6   output reg q
7       );
8       always @(posedge clk or posedge reset) begin
9       if(reset)//always at posivitive edge
10      q<=1'b1;
11      else if(t)
12      q<=~q;
13      end
14
15  endmodule
16
17  module counter_t_flip_flop(
18  input clk, reset, output [2:0]q
19      );
20      wire t0,t1,t2;
21      assign t0=(~q[1]) & (q[0] ^ q[2]);
22      assign t1=1;
23      assign t2=(q[1]) & (~(q[0] ^ q[2]));
24
25      t_ff ff0(t0,clk,reset,q[0]);
26      t_ff ff1(t1,clk,reset,q[1]);
27      t_ff ff2(t2,clk,reset,q[2]);
28
29  endmodule
```



Project Summary | Schematic | counter_lock.v | tb_counterLock.v

C:/Users/vinit/project_counterlockout/project_counterlockout.srcs/sim_1/new/tb_counterLock.v

```
1  `timescale 1ns / 1ps
2  module tb_counter(
3      );
4
5      reg clk,reset;
6      wire [2:0]q;
7
8      counter_t_flip_flop uut(clk,reset,q);
9      initial
10     begin
11     clk=0;
12     #5
13     forever #5 clk=~clk;
14     end
15
16     initial
17     begin
18     reset =1;
19     #10
20     reset=0;
21     #50
22     $finish;
23     end
24
25 endmodule
```