# DSD LAB SEE Verilog Codes (Xilinx)

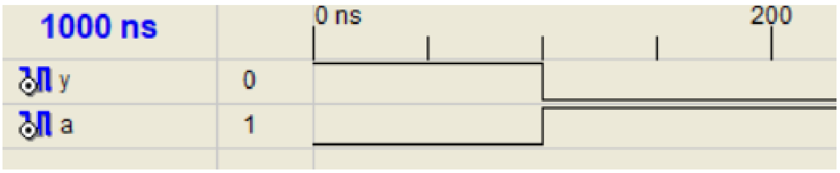
## 1.Code and simulate all Basic gates.

### NOT Gate:

- *Verilog Code:*

```
module not_gate(a, y);
    input a;
    output y;
    not G1(y,a);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*
  1. a=0;
  2. a=1;

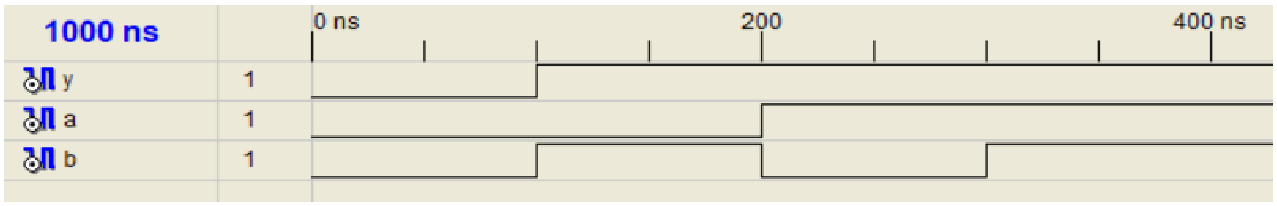


### OR Gate:

- *Verilog Code:*

```
module or_gate(a, b, y);
    input a;
    input b;
    output y;
    or G1(y,a,b);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*
  1. a=0;
     b=0;
  2. a=0;
     b=1;
  3. a=1;
     b=0;
  4. a=1;
     b=1;

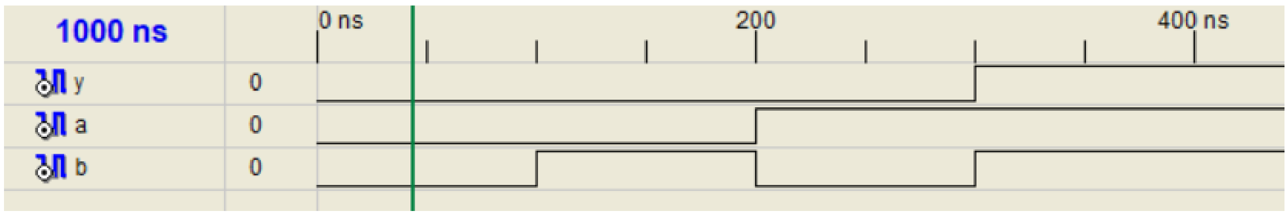


### AND Gate:

- *Verilog Code:*

```
module and_gate(a, b, y);
    input a;
    input b;
    output y;
    and G1(y,a,b);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*
  1. a=0;
     b=0;
  2. a=0;
     b=1;
  3. a=1;
     b=0;
  4. a=1;
     b=1;



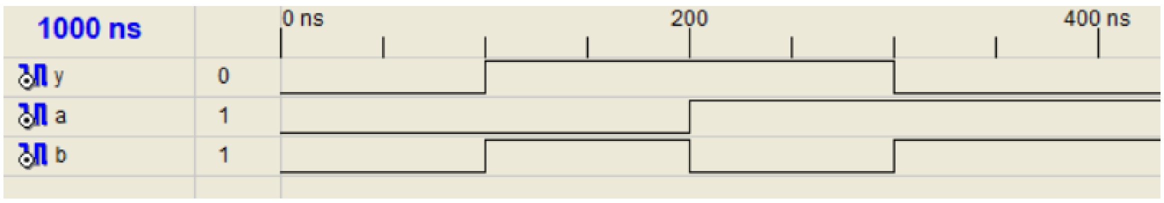
## XOR Gate:

- *Verilog Code:*

```
module xor_gate(a, b, y);
    input a;
    input b;
    output y;
    xor G1(y,a,b);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*

1. a=0;
   b=0;
2. a=0;
   b=1;
3. a=1;
   b=0;
4. a=1;
   b=1;



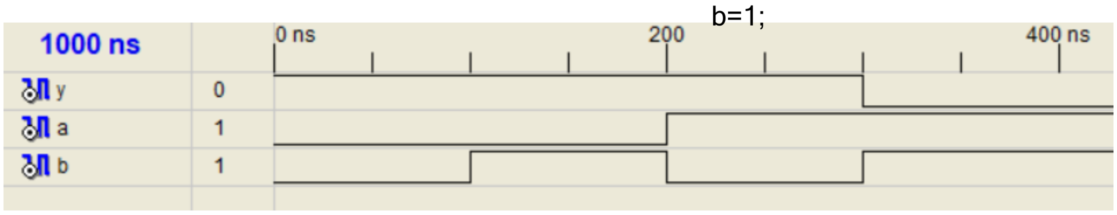
## NAND Gate:

- *Verilog Code:*

```
module nand_gate(a, b, y);
    input a;
    input b;
    output y;
    nand G1(y,a,b);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*

1. a=0;
   b=0;
2. a=0;
   b=1;
3. a=1;
   b=0;
4. a=1;
   b=1;



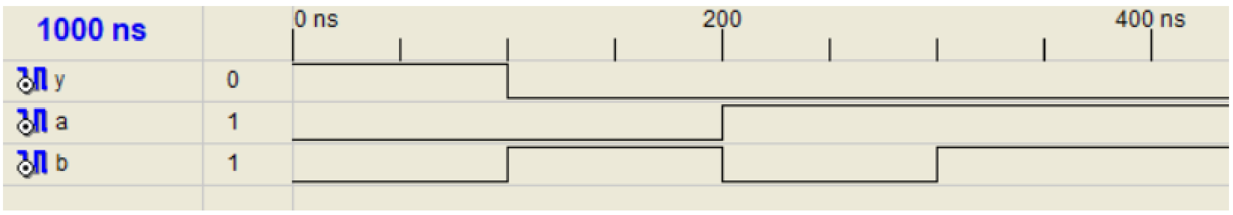
## NOR Gate:

- *Verilog Code:*

```
module nor_gate(a, b, y);
    input a;
    input b;
    output y;
    nor G1(y,a,b);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*

5. a=0;
   b=0;
6. a=0;
   b=1;
7. a=1;
   b=0;
8. a=1;
   b=1;



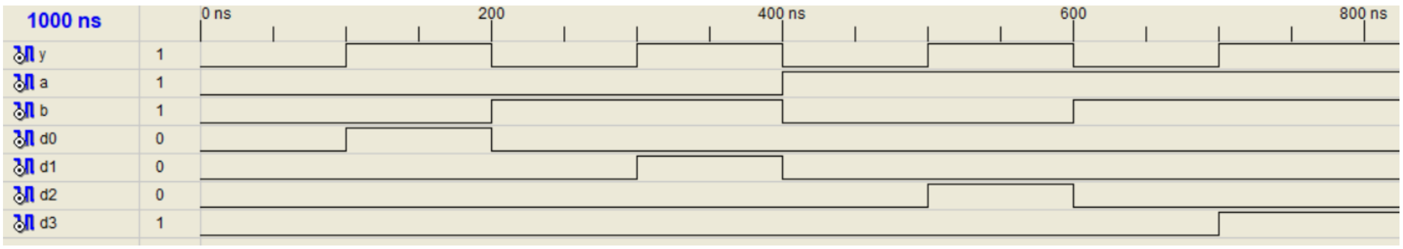
## 2.Simulation of 4:1 multiplexer using data flow model.

- *Verilog Code:*

```
module mux4to1(d0, d1, d2, d3, a, b, y);
     input d0;
     input d1;
     input d2;
     input d3;
     input a;
     input b;
     output y;
          assign
y=(~a&~b&d0)|(~a&b&d1)|(a&~b&d2)|(a&b&d3);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*

```
1. d0 = 0; d1 = 0; d2 = 0; d3 = 0; a = 0; b = 0;
2. d0 = 1; d1 = 0; d2 = 0; d3 = 0; a = 0; b = 0;
3. d0 = 0; d1 = 0; d2 = 0; d3 = 0; a = 0; b = 1;
4. d0 = 0; d1 = 1; d2 = 0; d3 = 0; a = 0; b = 1;
5. d0 = 0; d1 = 0; d2 = 0; d3 = 0; a = 1; b = 0;
6. d0 = 0; d1 = 0; d2 = 1; d3 = 0; a = 1; b = 0;
7. d0 = 0; d1 = 0; d2 = 0; d3 = 0; a = 1; b = 1;
8. d0 = 0; d1 = 0; d2 = 0; d3 = 1; a = 1; b = 1;
```



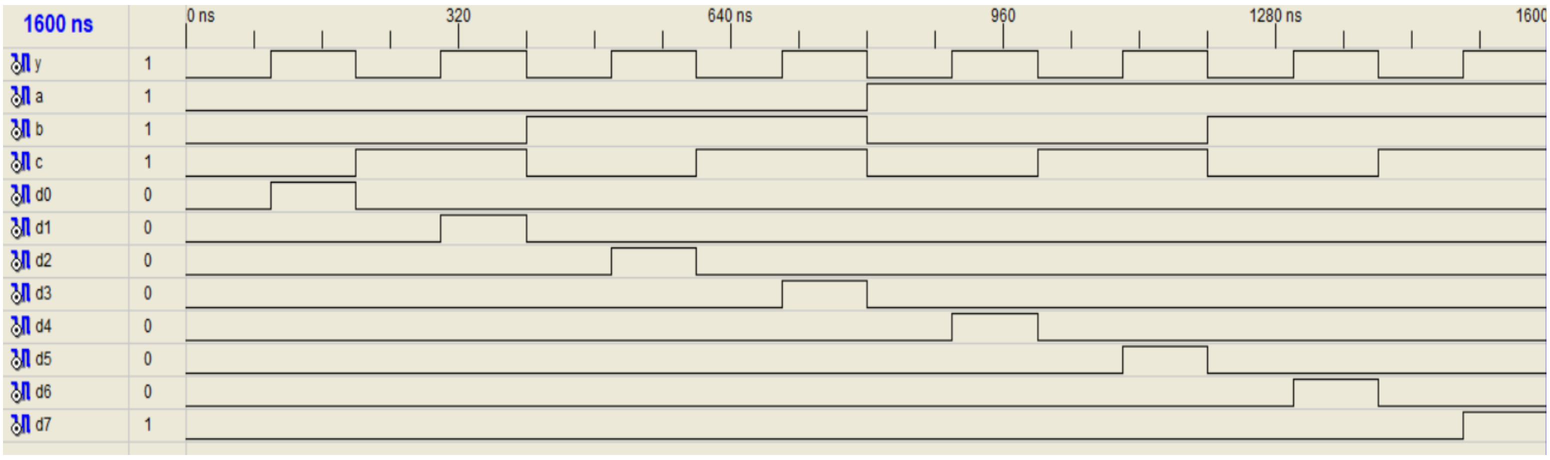
## 3.Simulation of 8:1 multiplexer.

- *Verilog Code:*

```
module mux8to1(d0, d1, d2, d3, d4, d5, d6, d7, a, b, c, y);
     input d0;
     input d1;
     input d2;
     input d3;
     input d4;
     input d5;
     input d6;
     input d7;
     input a;
     input b;
     input c;
     output y;
     reg y;
          always @(a or b or c or d0 or d1 or d2 or d3 or d4 or d5 or d6 or d7)
          case({a,b,c})
               0:y=d0;
               1:y=d1;
               2:y=d2;
               3:y=d3;
               4:y=d4;
               5:y=d5;
               6:y=d6;
               7:y=d7;
          endcase
endmodule
```

- *Verilog Test Fixture : (Input combinations)*

1. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 0; b = 0; c=0;
2. d0 = 1; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 0; b = 0; c=0;
3. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 0; b = 0; c=1;
4. d0 = 0; d1 = 1; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 0; b = 0; c=1;
5. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 0; b = 1; c=0;
6. d0 = 0; d1 = 0; d2 = 1; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 0; b = 1; c=0;
7. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 0; b = 1; c=1;
8. d0 = 0; d1 = 0; d2 = 0; d3 = 1; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 0; b = 1; c=1;
9. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 1; b = 0; c=0;
10. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 1; d5 = 0; d6 = 0; d7 = 0; a = 1; b = 0; c=0;
11. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 1; b = 0; c=1;
12. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 1; d6 = 0; d7 = 0; a = 1; b = 0; c=1;
13. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 1; b = 1; c=0;
14. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 1; d7 = 0; a = 1; b = 1; c=0;
15. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 0; a = 1; b = 1; c=1;
16. d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0; d6 = 0; d7 = 1; a = 1; b = 1; c=1;



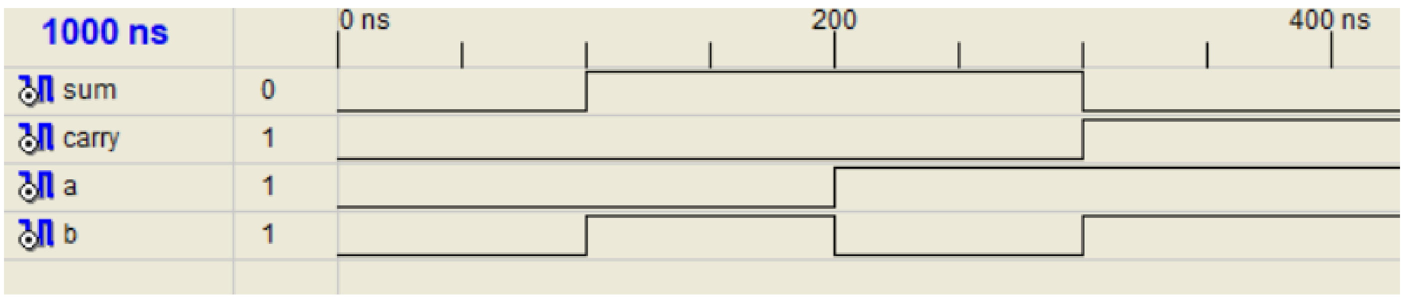
## 4.Simulation of Half adder using Data Flow model.

- *Verilog Code:*

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

- *Verilog Test Fixture : (Input combinations)*

1. a=0;
   b=0;
2. a=0;
   b=1;
3. a=1;
   b=0;
4. a=1;
   b=0;



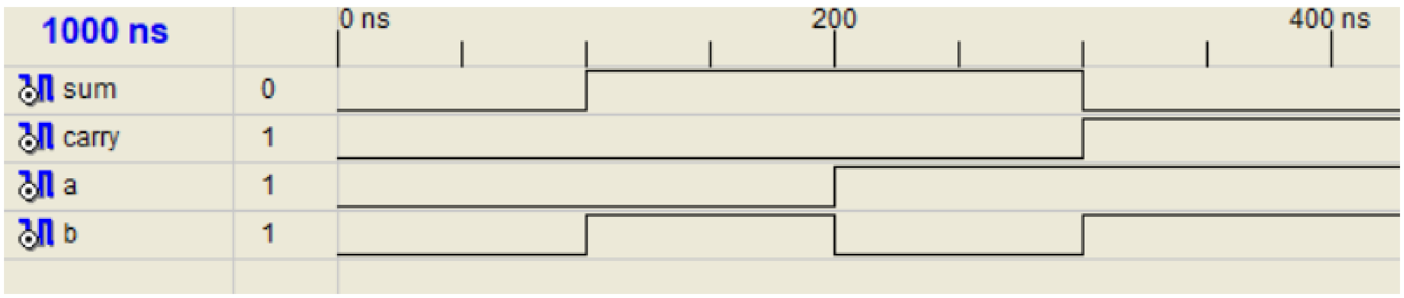
## 5.Simulation of Half adder using Behavioral model.

- *Verilog Code:*

```
module ha(a, b, sum, carry);
    input a;
    input b;
    output sum;
    output carry;
        reg sum;
        reg carry;
        always @(a or b)
        case({a,b})
              0: begin
                     sum=0; carry=0;
                 end
              1: begin
                     sum=1; carry=0;
                 end
              2: begin
                     sum=1; carry=0;
                 end
              3: begin
                     sum=0; carry=1;
                 end
       endcase
endmodule
```

- *Verilog Test Fixture : (Input combinations)*

1. a=0;
   b=0;
2. a=0;
   b=1;
3. a=1;
   b=0;
4. a=1;
   b=0;



## 6.Simulation of Full adder.

- *Verilog Code:*

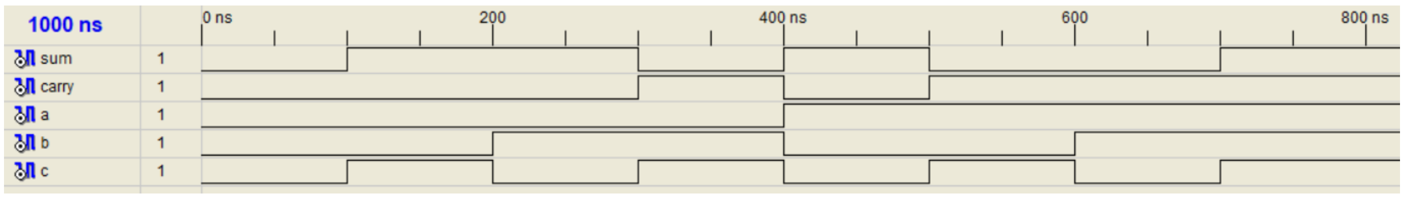
```
module full_adder(a, b, c, sum, carry);
    input a;
    input b;
    input c;
    output sum;
    output carry;
        reg sum;
        reg carry;
        always @(a or b or c)
    case({a,b,c})
       0: begin
              sum=0; carry=0;
          end
```

- *Verilog Test Fixture : (Input combinations)*

1. a=0;
   b=0;
   c=0;
2. a=0;
   b=0;
   c=1;
3. a=0;
   b=1;
   c=0;
4. a=0;
   b=1;
   c=1;
5. a=1;
   b=0;
   c=0;

```
        1:  begin
                sum=1;  carry=0;
            end
        2:  begin
                sum=1;  carry=0;
            end
        3:  begin
                sum=0;  carry=1;
            end
        4:  begin
                sum=1;  carry=0;
            end
        5:  begin
                sum=0;  carry=1;
            end
        6:  begin
                sum=0;  carry=1;
            end
        7:  begin
                sum=1;  carry=1;
            end
        endcase
endmodule
```

6. a=1;
   b=0;
   c=1;
7. a=1;
   b=1;
   c=1;
8. a=1;
   b=1;
   c=1;



## *7.Simulation of 1:8 demultiplexer.*

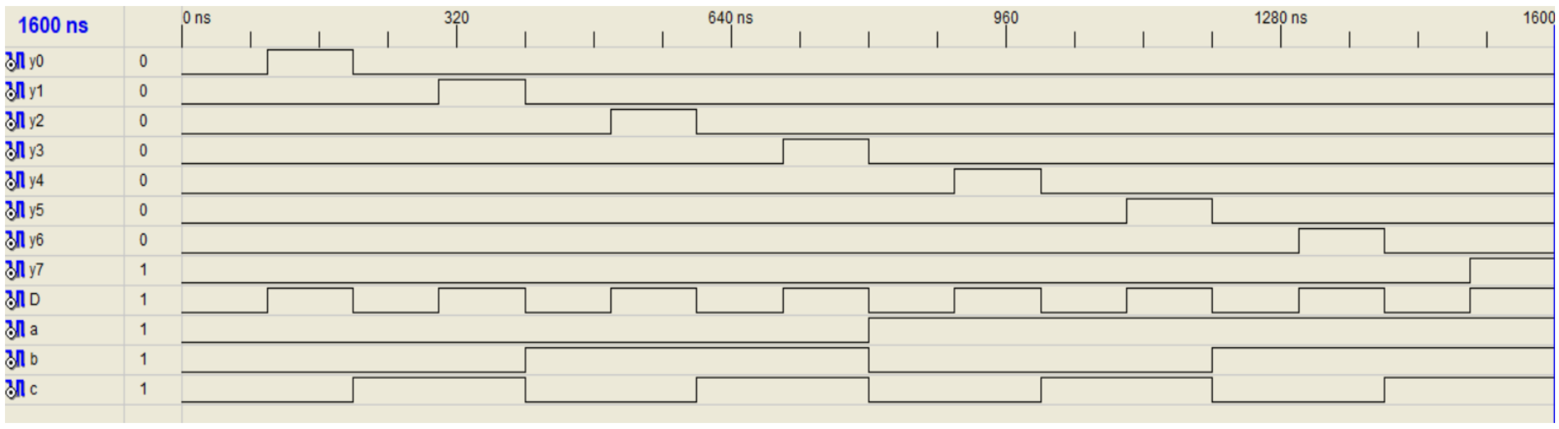
➢ *Verilog Code:*

```
module  demux1to8(a,  b,  c,  D,  y0,  y1,  y2,  y3,  y4,  y5,  y6,  y7);
    input  a;
    input  b;
    input  c;
    input  D;
        output  y0,y1,y2,y3,y4,y5,y6,y7;
        reg  y0,y1,y2,y3,y4,y5,y6,y7;
        always  @(a  or  b  or  c  or  D)
        case({a,b,c})
              0:begin
                     y0=D;  y1=0;  y2=0;  y3=0;  y4=0;  y5=0;  y6=0;  y7=0;
                end
              1:begin
                     y0=0;  y1=D;  y2=0;  y3=0;  y4=0;  y5=0;  y6=0;  y7=0;
                end
```

➢ *Verilog Test Fixture : (Input combinations)*

1. a=0; b=0; c=0; d=0;
2. a=0; b=0; c=0; d=1;
3. a=0; b=0; c=1; d=0;
4. a=0; b=0; c=1; d=1;
5. a=0; b=1; c=0; d=0;
6. a=0; b=1; c=0; d=1;
7. a=0; b=1; c=1; d=0;
8. a=0; b=1; c=1; d=1;
9. a=1; b=0; c=0; d=0;
10. a=1; b=0; c=0; d=1;
11. a=1; b=0; c=1; d=0;
12. a=1; b=0; c=1; d=1;
13. a=1; b=1; c=0; d=0;
14. a=1; b=1; c=0; d=1;
15. a=1; b=1; c=1; d=0;
16. a=1; b=1; c=1; d=1;

```
            2:begin
                    y0=0;  y1=0;  y2=D;  y3=0;  y4=0;  y5=0;  y6=0;  y7=0;
              end
            3:begin
                    y0=0;  y1=0;  y2=0;  y3=D;  y4=0;  y5=0;  y6=0;  y7=0;
              end
            4:begin
                    y0=0;  y1=0;  y2=0;  y3=0;  y4=D;  y5=0;  y6=0;  y7=0;
              end
            5:begin
                    y0=0;  y1=0;  y2=0;  y3=0;  y4=0;  y5=D;  y6=0;  y7=0;
              end
            6:begin
                    y0=0;  y1=0;  y2=0;  y3=0;  y4=0;  y5=0;  y6=D;  y7=0;
              end
            7:begin
                    y0=0;  y1=0;  y2=0;  y3=0;  y4=0;  y5=0;  y6=0;  y7=D;
              end
    endcase
endmodule
```



## 8.Simulation of 3:8 decoder.

➢ *Verilog Code:*

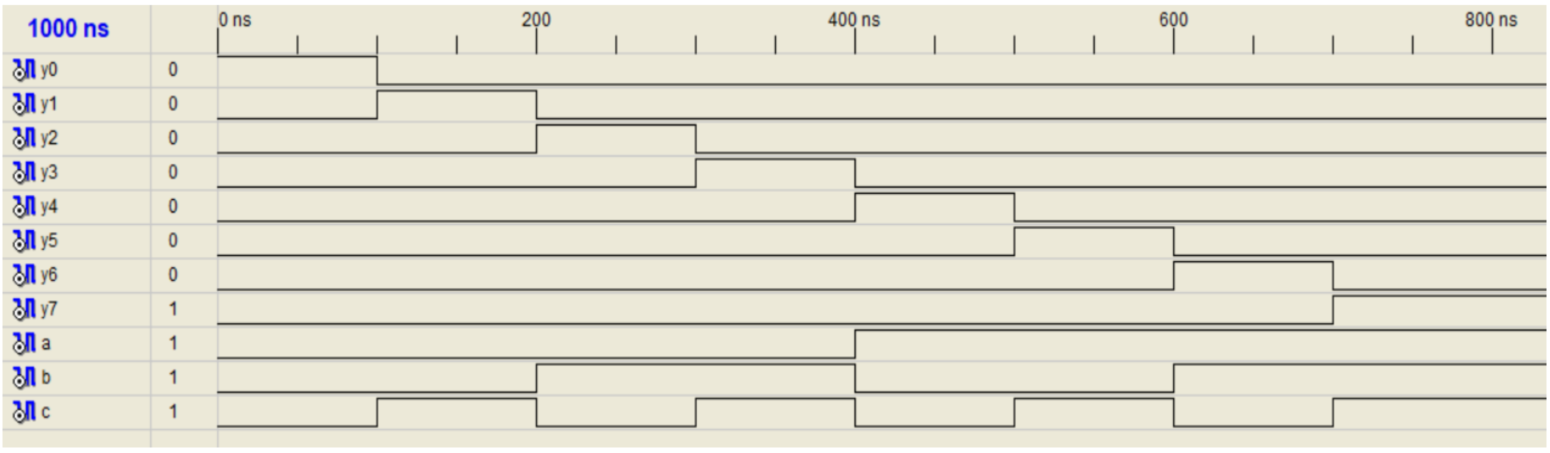
```
module decoder3to8(a, b, c, y0, y1, y2, y3, y4, y5, y6, y7);
    input a;
    input b;
    input c;
    output y0,y1,y2,y3,y4,y5,y6,y7;
        reg y0,y1,y2,y3,y4,y5,y6,y7;
        always @(a or b or c)
        case({a,b,c})
        0:  begin
                    y0=1;  y1=0;  y2=0;  y3=0;  y4=0;  y5=0;  y6=0;  y7=0;
              end
        1:  begin
                    y0=0;  y1=1;  y2=0;  y3=0;  y4=0;  y5=0;  y6=0;  y7=0;
              end
```

➢ *Verilog Test Fixture : (Input combinations)*

1. a=0;
   b=0;
   c=0;
2. a=0;
   b=0;
   c=1;
3. a=0;
   b=1;
   c=0;
4. a=0;
   b=1;
   c=1;
5. a=1;
   b=0;
   c=0;

```
    2:  begin
                    y0=0;  y1=0;  y2=1;  y3=0;  y4=0;  y5=0;  y6=0;  y7=0;
            end
    3:  begin
                    y0=0;  y1=0;  y2=0;  y3=1;  y4=0;  y5=0;  y6=0;  y7=0;
            end
    4:  begin
                    y0=0;  y1=0;  y2=0;  y3=0;  y4=1;  y5=0;  y6=0;  y7=0;
            end
    5:  begin
                    y0=0;  y1=0;  y2=0;  y3=0;  y4=0;  y5=1;  y6=0;  y7=0;
            end
    6:  begin
                    y0=0;  y1=0;  y2=0;  y3=0;  y4=0;  y5=0;  y6=1;  y7=0;
            end
    7:  begin
                    y0=0;  y1=0;  y2=0;  y3=0;  y4=0;  y5=0;  y6=0;  y7=1;
            end
   endcase
endmodule
```

6. a=1;
   b=0;
   c=1;
7. a=1;
   b=1;
   c=1;
8. a=1;
   b=1;
   c=1;



## 9.Simulation of flip flops : i)D flip flop  ii)RS flip flop

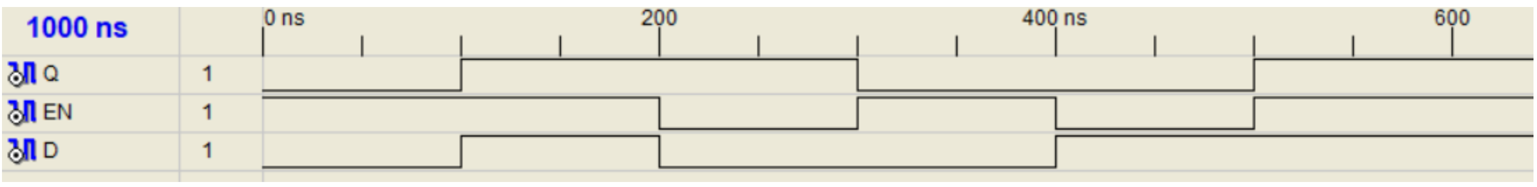
### i)a.Gated D Flip Flop:

➢ *Verilog Code:*

```
module  gated_D_FF(D,  EN,  Q);
      input  D;
      input  EN;
      output  Q;
            reg  Q;
            always   @(D  or  EN)
            if(EN)
                  Q=D;
endmodule
```

➢ *Verilog Test Fixture : (Input combinations)*

1. D=0; EN=1;
2. D=1; EN=1;
3. D=0; EN=0;
4. D=0; EN=1;
5. D=1; EN=0;
6. D=1; EN=1;

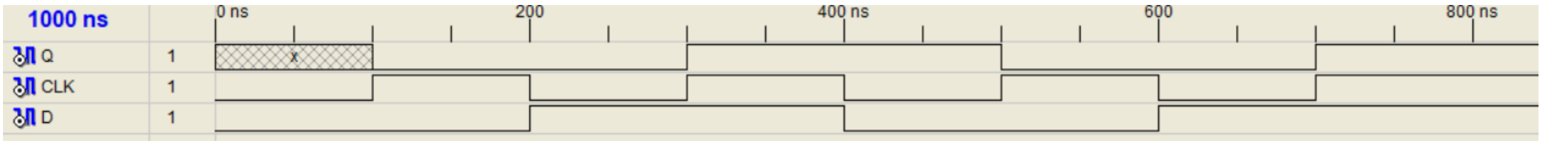


**i)b.Edge Triggered D Flip Flop:**

- *Verilog Code:*

```
module edge_triggered_D_FF(D, CLK, Q);
    input D;
    input CLK;
    output Q;
        reg Q;
        always @(posedge CLK)
              Q=D;
endmodule
```

- *Verilog Test Fixture : (Input combinations)*
  1. D=0; CLK=0;
  2. D=0; CLK=1;
  3. D=1; CLK=0;
  4. D=1; CLK=1;
  5. D=0; CLK=0;
  6. D=0; CLK=1;
  7. D=1; CLK=0;
  8. D=1; CLK=1;

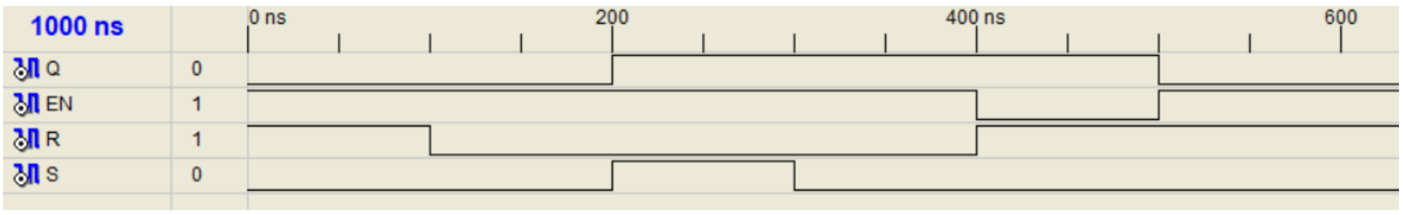


**ii)a.Gated RS Flip Flop:**

- *Verilog Code:*

```
module gated_RS_FF(R,S, EN, Q);
    input R;
        input S;
    input EN;
    output Q;
        reg Q;
        always  @(R or S or EN)
        if(EN)
              Q=S|(~R&Q);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*
  1. R=1; S=0; EN=1;
  2. R=0; S=0; EN=1;
  3. R=0; S=1; EN=1;
  4. R=0; S=0; EN=1;
  5. R=1; S=0; EN=0;
  6. R=1; S=0; EN=1;

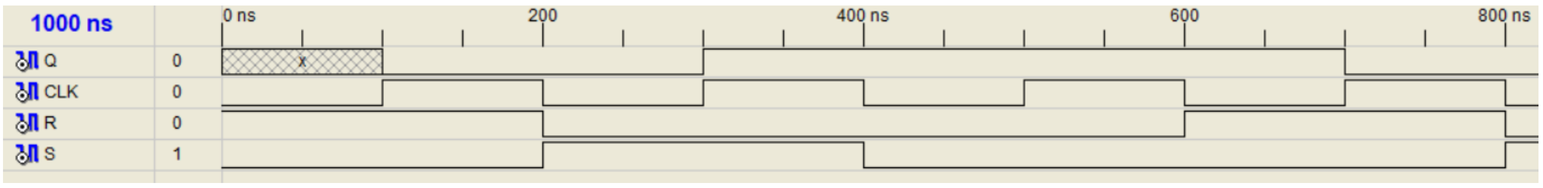


**ii)b.Edge Triggered RS Flip Flop:**

- *Verilog Code:*

```
module edge_triggered_RS_FF(R,S, CLK, Q);
    input R;
        input S;
    input CLK;
    output Q;
        reg Q;
        always  @(posedge CLK)
              Q=S|(~R&Q);
endmodule
```

- *Verilog Test Fixture : (Input combinations)*
  1. R=1; S=0; CLK=0;
  2. R=1; S=0; CLK=1;
  3. R=0; S=1; CLK=0;
  4. R=0; S=1; CLK=1;
  5. R=0; S=0; CLK=0;
  6. R=0; S=0; CLK=1;
  7. R=1; S=0; CLK=0;
  8. R=1; S=0; CLK=1;
  9. R=0; S=1; CLK=0;



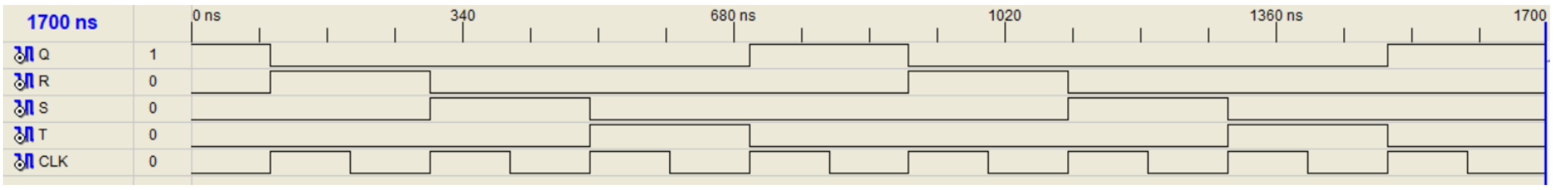
## *10.Simulation of Ring counter*

➢ *Verilog Code:*

```
module ring_counter(CLK, Q, R, S, T);
      input CLK;
      output Q,R,S,T;
      reg Q,R,S,T;
            initial
                  begin
                        Q=1; R=0; S=0; T=0;
                  end
            always @(posedge CLK)
                  begin
                        Q<=T; R<=Q; S<=R; T<=S;
                  end
endmodule
```

➢ *Verilog Test Fixture : (Input combinations)*

1. CLK=0;
2. CLK=1;
3. CLK=0;
4. CLK=1;
5. CLK=0;
6. CLK=1;
7. CLK=0;
8. CLK=1;
9. CLK=0;
10. CLK=1;
11. CLK=0;
12. CLK=1;
13. CLK=0;
14. CLK=1;
15. CLK=0;
16. CLK=1;
17. CLK=0;



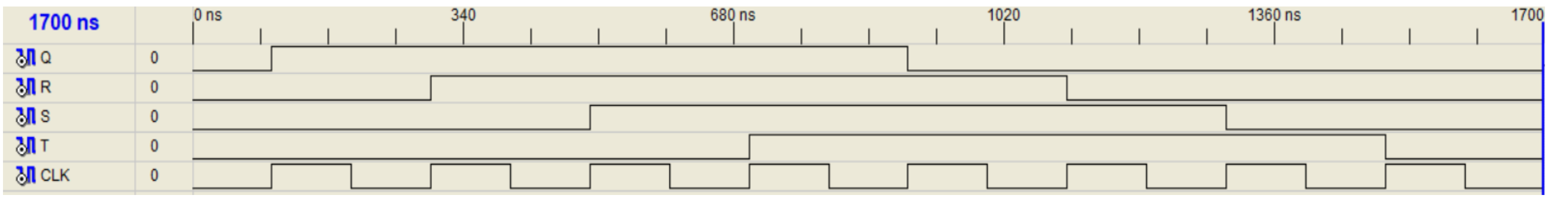
## *11.Simulation of Johnson counter*

➢ *Verilog Code:*

```
module johnson_counter(CLK, Q, R, S, T);
      input CLK;
      output Q,R,S,T;
      reg Q,R,S,T;
            initial
                  begin
                        Q=0; R=0; S=0; T=0;
                  end
            always @(posedge CLK)
                  begin
                        Q<=~T; R<=Q; S<=R; T<=S;
                  end
endmodule
```

➢ *Verilog Test Fixture : (Input combinations)*

1. CLK=0;
2. CLK=1;
3. CLK=0;
4. CLK=1;
5. CLK=0;
6. CLK=1;
7. CLK=0;
8. CLK=1;
9. CLK=0;
10. CLK=1;
11. CLK=0;
12. CLK=1;
13. CLK=0;
14. CLK=1;
15. CLK=0;
16. CLK=1;
17. CLK=0;



***