ज्ञानेन प्रकाशते जगत्
INDUS UNIVERSITY

Digital System Design using HDL(EC0419)
Unit-2
B.Tech (Electronics and Communication)
Semester-IV

Faculty Name : Prof. Miloni Ganatra

Academic Year 2020

Unit 2: Gate Level Modeling

Gate Level Modeling:
Introduction, Gate level Primitive, Module Structure, Instances of Primitives, Gate Delays, Designing Using Primitives.

Gate Level Modeling

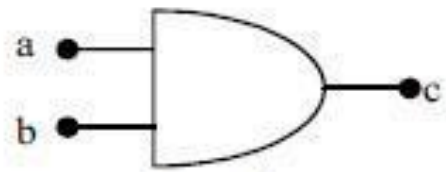
- Steps
  - Develope the boolean function of output
  - Draw the circuit with logic gates/primitives
  - Connect gates/primitives with net (usually wire)
- HDL: Hardware **Description** Language
  - Figure out architecture first, then write code.

Gate Level

- At the next higher level of abstraction,

  design is carried out in terms of basic gates.

- All the basic gates are available as ready modules called "*Primitives*".



c = a . b

Primitives

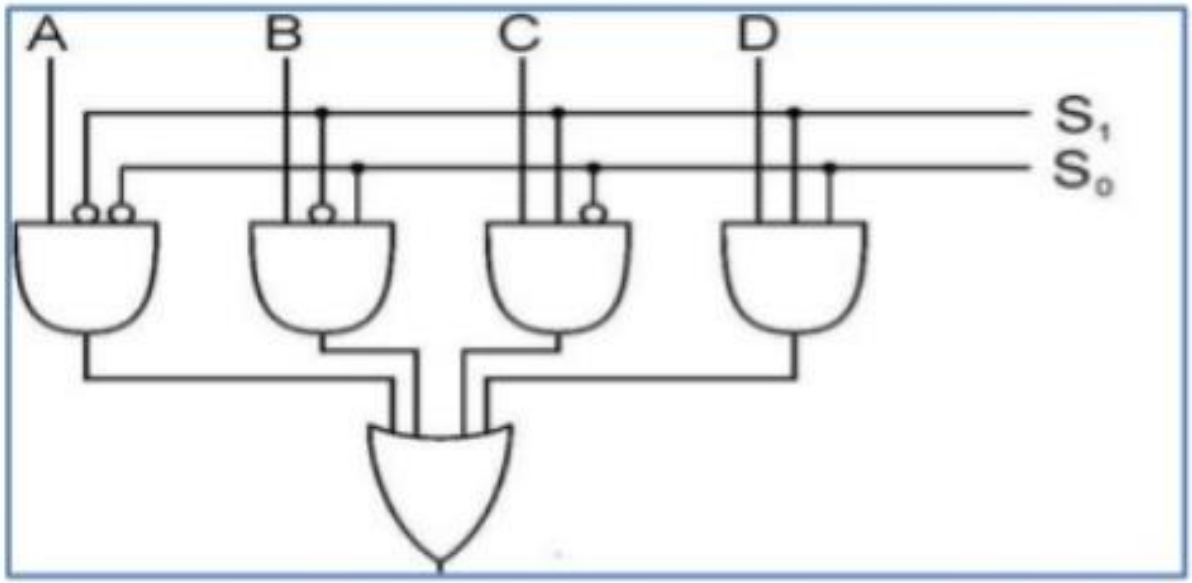
- Primitives are modules ready to be instanced
- Smallest modeling block for simulator
- Verilog build-in primitive gate
  - *and, or, not, buf, xor, nand, nor, xnor*
  - prim_name inst_name(output, in0, in1,....);
- User defined primitive (UDP)
  - building block defined by designer

Gate primitives

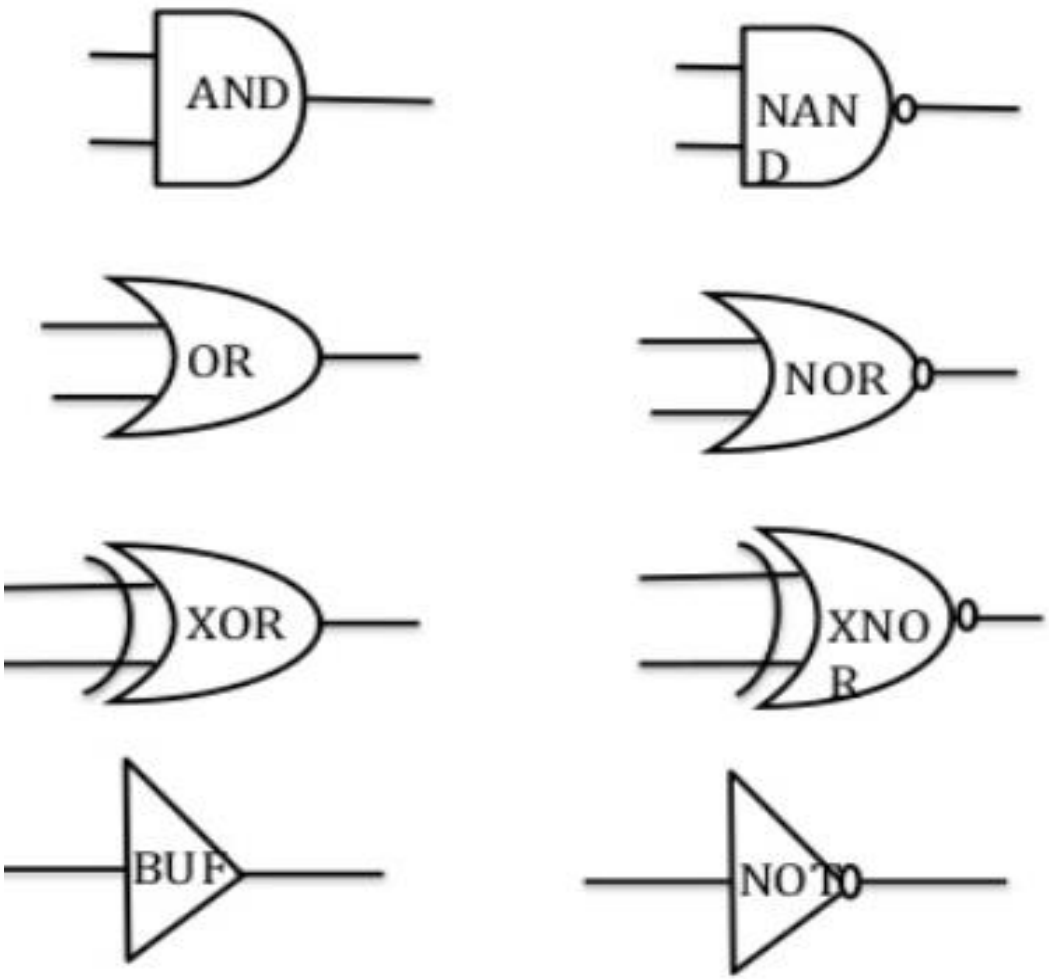
- Gate primitives are predefined in Verilog, which are ready to use.
- They are instantiated like modules. There are two classes of gate primitives:
- Multiple input gate primitives and Single input gate primitives. Multiple input gate primitives include and, nand, or, nor, xor, and xnor. These can have multiple inputs and a single output.
- Single input gate primitives include not, buf

Gate level Abstraction

- This is also known as structural level Abstraction.
- In this level the Design is described in terms of gates.
- It is very easy to design any circuit in verilog if we have the structure.
- For large circuit its difficult to implement in gate level.



Basic gate primitive in verilog



Instantiation of gates

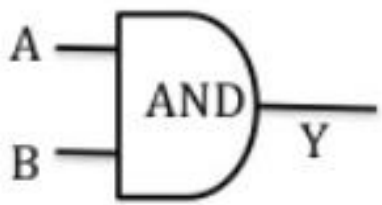
```
input wire   a, b ;
output wire y;
and  a1 ( y , a, b);
nand n1 ( y , a, b);
or o1 ( y , a, b);
nor no1  ( y , a, b);
xor x1 ( y , a, b);
xnor xn1 ( y , a, b);
buf b1 ( out,in);
not n1 (out,in);
```

GATE LEVEL MODELING

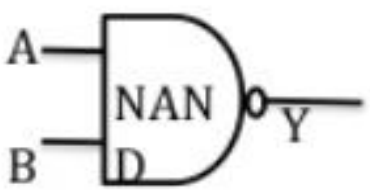
- All the basic gates are available as “Primitives” in Verilog.

Gate	Mode of instantiation	Output port(s)	Input port(s)
AND	`and` ga (o, i1, i2, . . . i8);	o	i1, i2, . .
OR	`or` gr (o, i1, i2, . . . i8);	o	i1, i2, . .
NAND	`nand` gna (o, i1, i2, . . . i8);	o	i1, i2, . .
NOR	`nor` gnr (o, i1, i2, . . . i8);	o	i1, i2, . .
XOR	`xor` gxr (o, i1, i2, . . . i8);	o	i1, i2, . .
XNOR	`xnor` gxn (o, i1, i2, . . . i8);	o	i1, i2, . .
BUF	`buf` gb (o1, o2, ... i);	o1, o2, o3, . .	i
NOT	`not` gn (o1, o2, o3, . . . i);	o1, o2, o3, . .	i

Basic gate primitive in verilog



AND	0	1	X	Z
0	0	0	0	0
1	0	1	X	X
X	0	X	X	X
Z	0	X	X	X



NAND	0	1	X	Z
0	1	1	1	1
1	1	0	X	X
X	1	X	X	X
Z	1	X	X	X

Basic gate primitive in verilog

OR	0	1	X	Z
0	0	1	X	X
1	1	1	1	1
X	X	1	X	X
Z	X	1	X	X

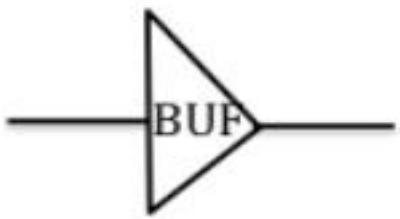
NOR	0	1	X	Z
0	1	0	X	X
1	0	0	0	0
X	X	0	X	X
Z	X	0	X	X

Basic gate primitive in verilog

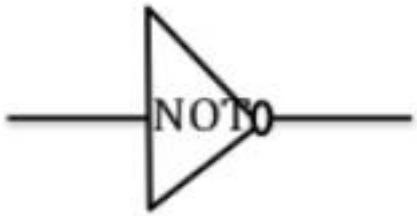
XOR	0	1	X	Z
0	0	1	X	X
1	1	0	X	X
X	X	X	X	X
Z	X	X	X	X

XNOR	0	1	X	Z
0	1	0	X	X
1	0	1	X	X
X	X	X	X	X

Basic gate primitive in verilog



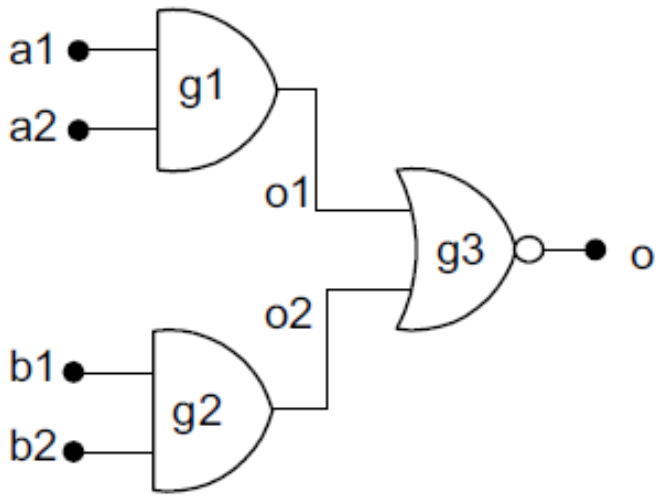
Buf	in	out
	0	0
	1	1
	X	X
	Z	X



Buf	in	out
	0	1
	1	0
	X	X
	Z	X

Verilog module for AOI logic

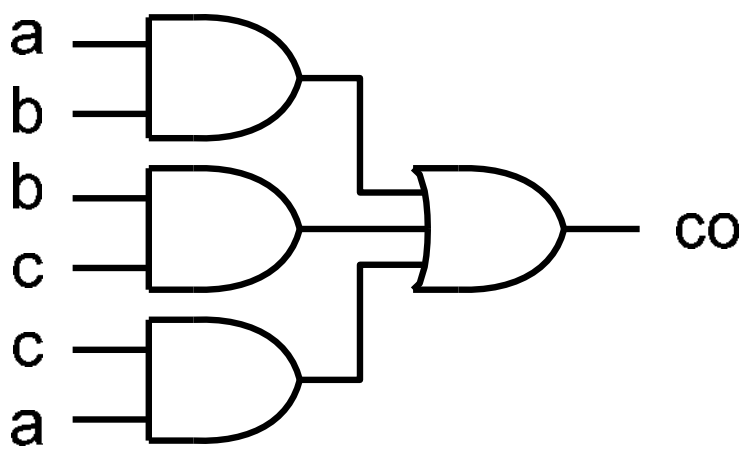
- module aoi_gate(o,a1,a2,b1,b2);
  input a1,a2,b1,b2; output o;
  wire o1,o2;
  and g1(o1,a1,a2);
  and g2(o2,b1,b2);
  nor g3(o,o1,o2);
  endmodule



- module aoi_st;
  reg a1,a2,b1,b2;
  wire o;
  initial
  begin
  a1 = 0; a2 = 0; b1 = 0; b2 = 0;
  #3 a1 = 1; a2 = 1; b1 = 1; b2 = 0;
  end
  initial #100 $stop;
  initial $monitor($time , " o = %b , a1 = %b , a2 = %b , b1 = %b ,b2 = %b ",o,a1,a2,b1,b2);
  aoi_gate gg(o,a1,a2,b1,b2);
  endmodule

Case Study

- co = (a • b) + (b • c) + (c • a);




```
30
31 module FA_co ( co, a, b, c
32
33   input   a, b, c ;
34   output  co;
35   wire    ab, bc, ca;
36
37   and g0( ab, a, b );
38   and g1( bc, b, c );
39   and g2( ca, c, a );
40   or  g3( co, ab, bc, ca );
41
42 endmodule
43
```

Example of verilog Design using gate level

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- I/P => A , B
- O/P => Sum & carry

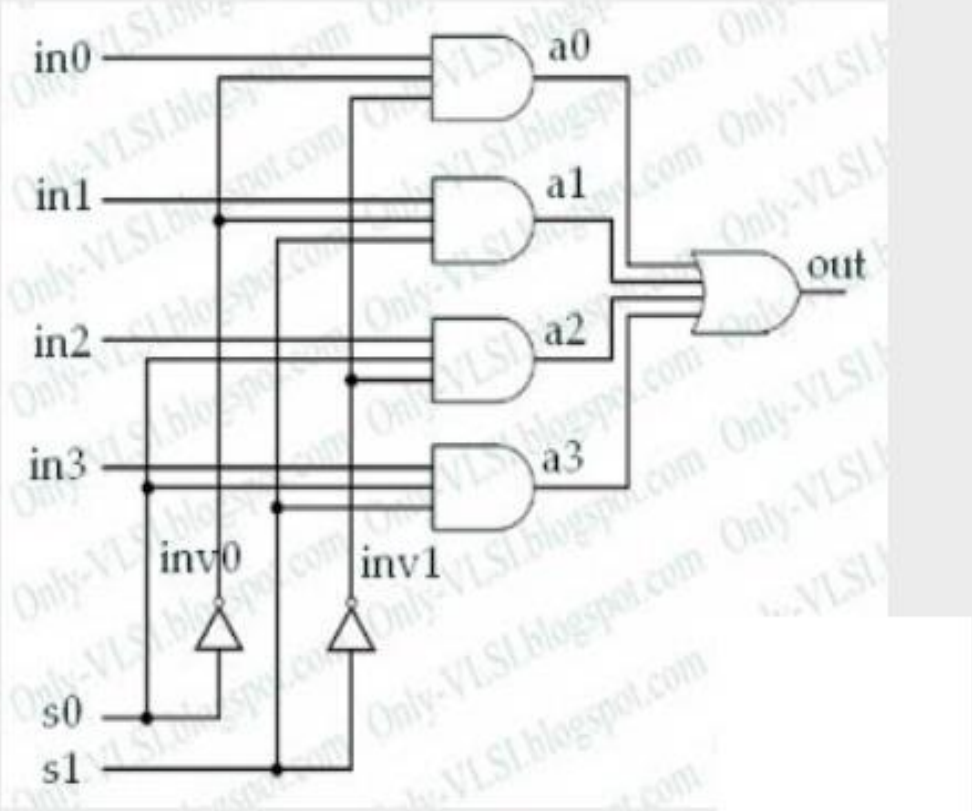
- Boolean Function
- Sum = A^B
- Carry = A.B

Example of verilog Design using gate level

```
// this is half adder verilog code
module half_adder (S, C , A, B )    // module name & port declaration

   output S;                        // output port declarations
   output C;                        // output port declarations

   input A;                         // input port declarations
   input B;                         // input port declarations

  xor  x1 ( S, A, B);               // instantiation of xor gate
  and  a1 (C, A, B);                // instantiation of and gate

endmodule
```

Gate level modeling of a 4x1 multiplexer.

The gate-level circuit diagram of 4x1 mux is shown below. It is used to write a module for 4x1 mux.

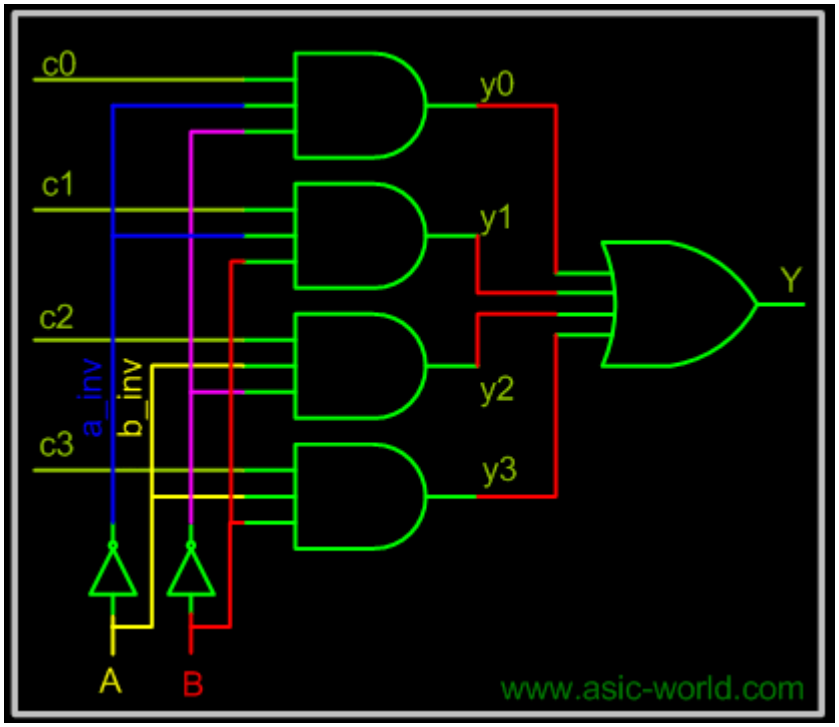


```
module 4x1_mux (out, in0, in1, in2, in3, s0, s1)

// port declarations
output out; // Output port.
input in0, in1, in2. in3; // Input ports.
input s0, s1; // Input ports: select lines.

// intermediate wires
wire inv0, inv1; // Inverter outputs.
wire a0, a1, a2, a3; // AND gates outputs.

// Inverters.
not not_0 (inv0, s0);
not not_1 (inv1, s1);
```

```
// 3-input AND gates.
and and_0 (a0, in0, inv0, inv1);
and and_1 (a1, in1, inv0, s1);
and and_2 (a2, in2, s0, inv1);
and and_3 (a3, in3, s0, s1);

// 4-input OR gate.
or or_0 (out, a0, a1, a2, a3);

endmodule
```

Simulation of 4-1 Mux using Gate level

```
// Testbench Code goes here
initial begin
                (

   c0, c1, c2, c3, A, B, Y);
  c0 = 0;
  c1 = 0;
  c2 = 0;
  c3 = 0;
  A = 0;
  B = 0;
  #1  A  = 1;
  #2  B  = 1;
  #4  A  = 0;
  #8  $finish;
end
```



ARRAY OF INSTANCES OF PRIMITIVES

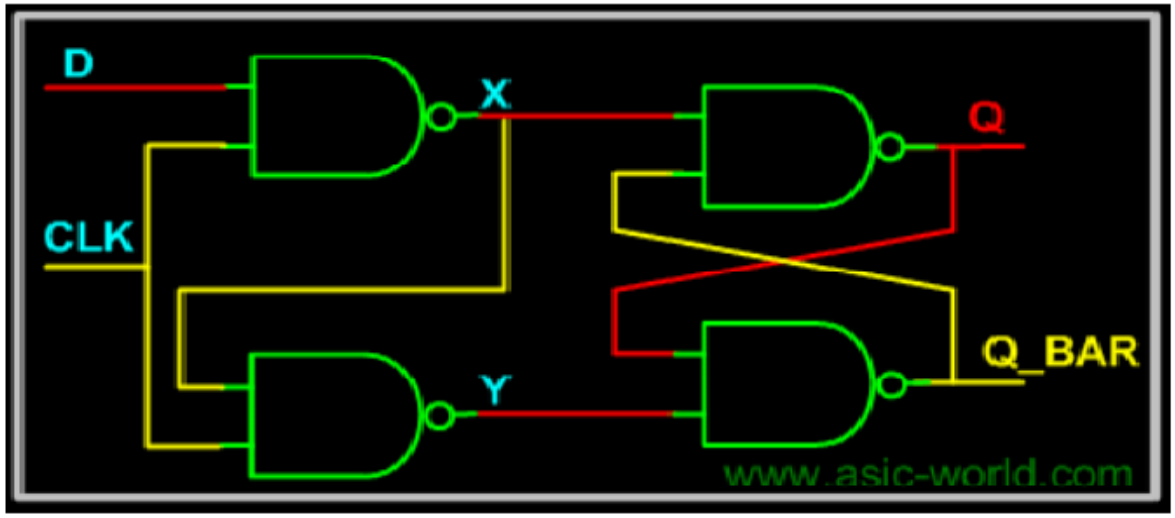
- and gate [7 : 4] (a, b, c);

- and gate [7] (a[3], b[3], c[3]),
  gate [6] (a[2], b[2], c[2]),
  gate [5] (a[1], b[1], c[1]),
  gate [4] (a[0], b[0], c[0]);

Syntax: **and gate[mm : nn](a, b, c);**

D-Flip flop from NAND Gate (Gate Level)

D-Flip flop from NAND Gate



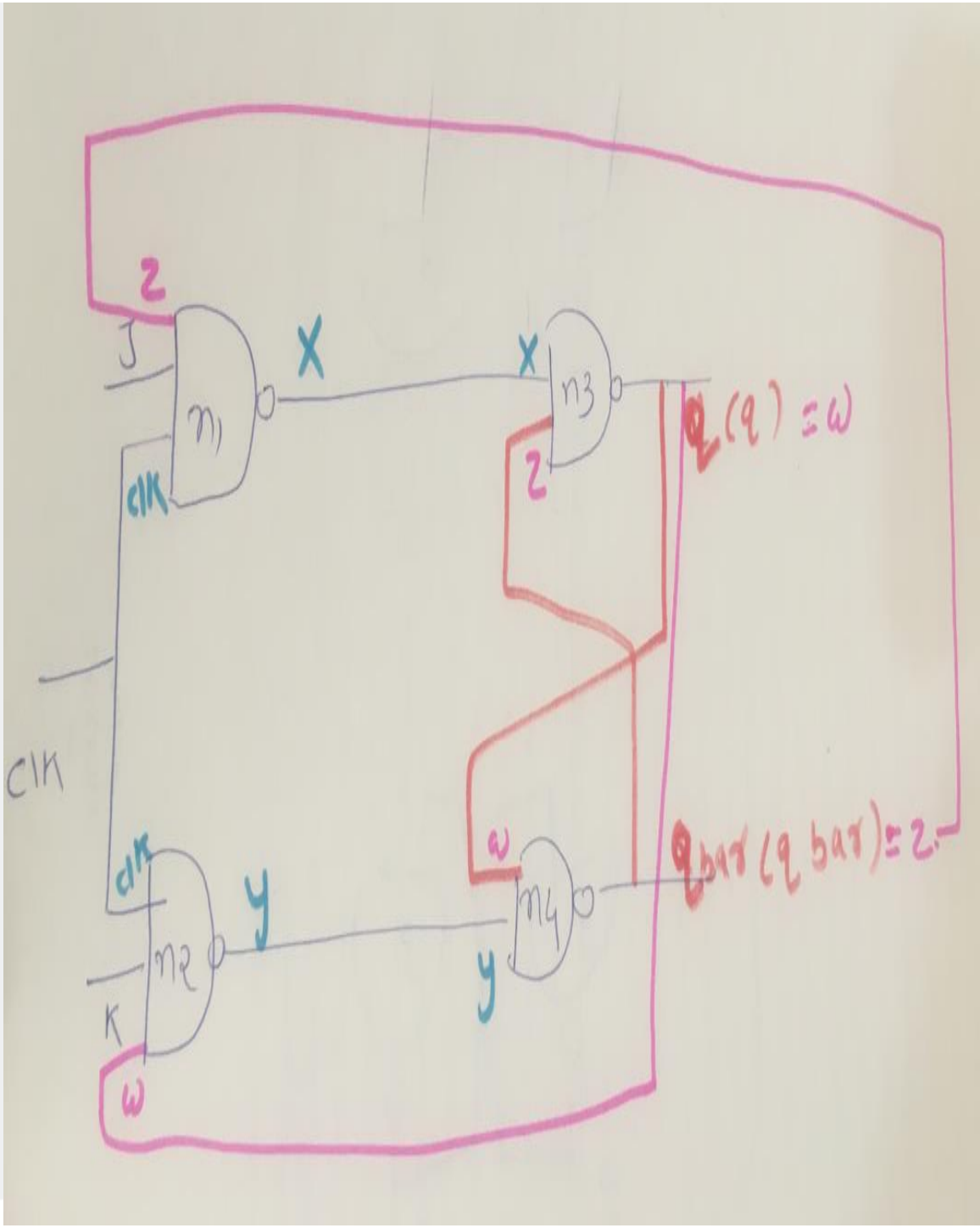
Verilog Code

```
module dff_from_nand();
wire Q,Q_BAR;
reg D,CLK;

nand U1 (X,D,CLK) ;
nand U2 (Y,X,CLK) ;
nand U3 (Q,Q_BAR,X);
nand U4 (Q_BAR,Q,Y);

```

```
// Testbench of above code
initial begin

  CLK = 0;
  D = 0;
  #3  D = 1;
  #3  D = 0;

end
```

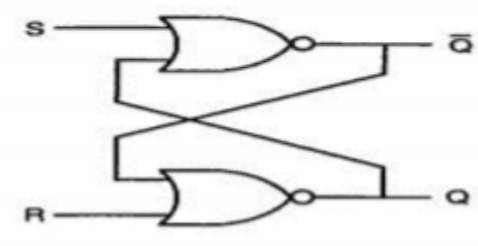
JK flip flop using gate level

```
module jkstruct(j,k,clk,q,qbar);

input j,k,clk;

output reg q,qbar;

initial begin q=1'b1;qbar=1'b0; end

wire x,y,w,z;

assign w=q;

assign z=qbar;

nand n1(x,z,j,clk);

nand n2(y,k,w,clk);

nand n3(q,x,z);

nand n4(qbar,y,w);

endmodule
```



SR latch using gate level

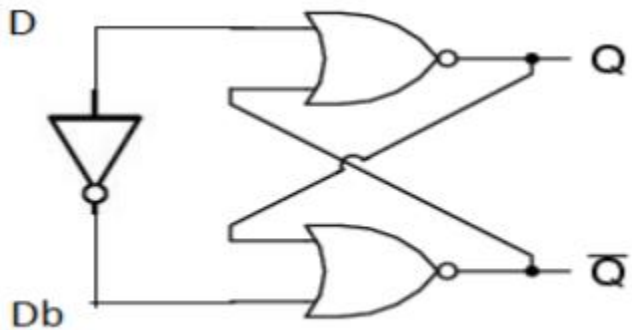
LOGIC DIAGRAM



VERILOG CODE

```
module srg(s,r,q,qb);
   input s;
   input r;
   inout q;
   inout qb;
   nor n1(qb,s,q);
   nor n2(q,r,qb);
endmodule
```

D latch using gate level




```
module dlatch(d,db,q,qb);
 input d;
input db;
inout q;
inout qb;
not (db,d);
nor n1(q,d,qb);
nor n2(qb,db,q);
 endmodule
```

Assignment

- **Write a Verilog code for following digital circuit using gate level modeling.**
- RS flip flop
- JK latch
- 3-8 Decoder
- 4-2 Encoder
- 8-1 Multiplexer
- 1-4 Demultiplexer