Phần bài tập

HTSLT3

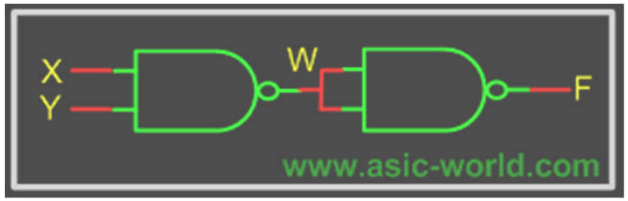
Thiết kế dùng cổng

- Chỉ dùng khi xây dựng thư viện
- Nhà cung cấp ASIC sẽ cung cấp thư viện ASIC Verilog dùng các cổng cơ bản
- Và người dùng có thể tự định nghĩa các cổng (mạch) - UDP

- Yêu cầu sinh viên
  - Bài 1: Xâu dựng cổng AND từ cổng NAND
  - Bài 2: Xây dựng D-FF từ cổng NAND
  - Bài 3: Xây dựng bộ cộng 1 bit và bộ cộng 4 bit
  - Bài 4: Xây dựng bộ ghép kênh 4:1
  - Bài 5: Xây dựng bộ so sánh 8 bit
  - Bài 6: Xây dựng bộ chuyển mã BCD

Bài 1: Xây dựng cổng AND từ cổng NAND

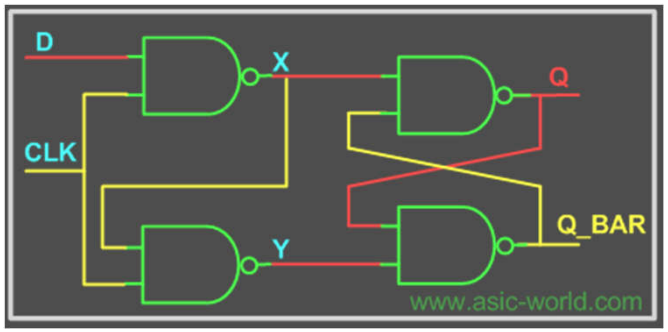
```
1  // Structural model of AND gate from two NANDS
2  module and_from_nand();
3
4  reg X, Y;
5  wire F, W;
6  // Two instantiations of the module NAND
7  nand U1(W,X, Y);
8  nand U2(F, W, W);
9
10 // Testbench Code
11 initial begin
12    $monitor ("X = %b Y = %b F = %b", X, Y, F);
13    X = 0;
14    Y = 0;
15    #1  X = 1;
16    #1  Y = 1;
17    #1  X = 0;
18    #1  $finish;
19 end
20
21 endmodule
```



X = 0 Y = 0 F = 0
X = 1 Y = 0 F = 0
X = 1 Y = 1 F = 1
X = 0 Y = 1 F = 0

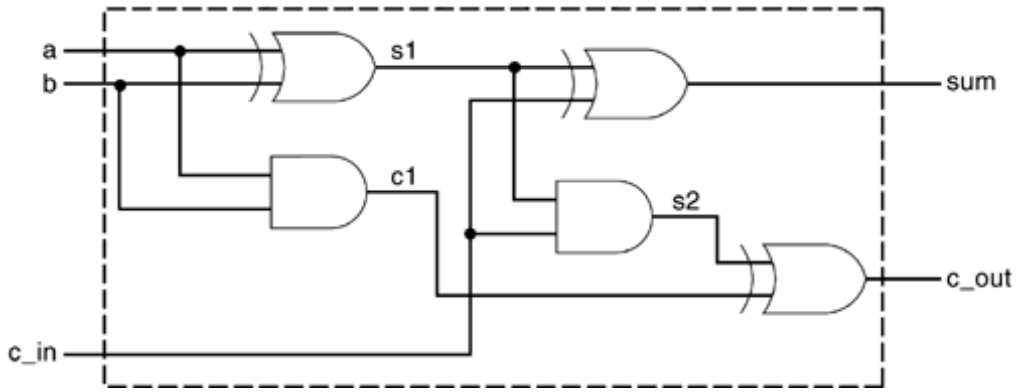
Bài 2: Xây dựng cổng D-FF từ cổng NAND

```
1  module dff_from_nand();
2  wire Q,Q_BAR;
3  reg D,CLK;
4
5  nand U1 (X,D,CLK) ;
6  nand U2 (Y,X,CLK) ;
7  nand U3 (Q,Q_BAR,X);
8  nand U4 (Q_BAR,Q,Y);
9
10 // Testbench of above code
11 initial begin
12    $monitor("CLK = %b D = %b Q = %b Q_BAR = %b",CLK, D, Q, Q_BAR);
13    CLK = 0;
14    D = 0;
15    #3  D = 1;
16    #3  D = 0;
17    #3  $finish;
18 end
19
20 always #2  CLK = ~CLK;
21
22 endmodule
```



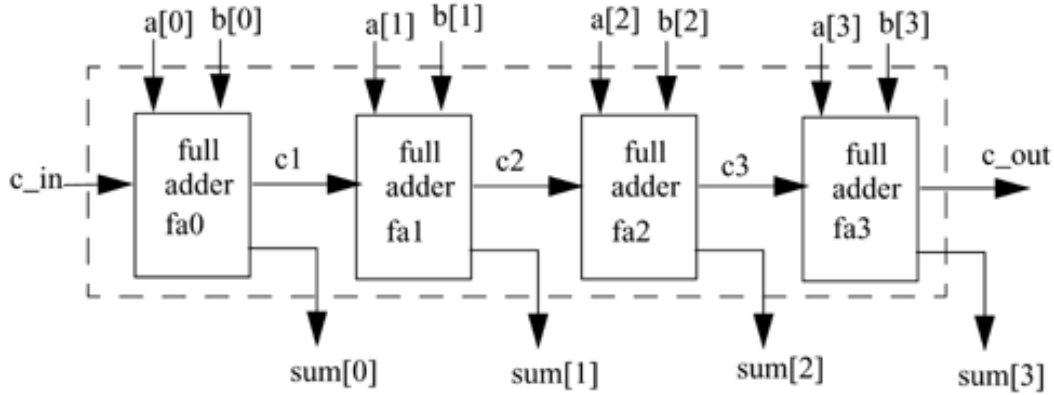
Bài 3: Bộ cộng đầy đủ 1 bit và 4 bit

```
module fulladder (sum,c_out,a, b, c_in);
//khai báo cổng vào ra

output sum,c_out;
input a, b, c_in;
wire s1, c1, c2;

//khai báo cổng logic
xor (s1,a,b);
and (c1,a,b);
xor (sum,s1,c_in);
and (s2,s1,c_in);
xor (c_out, s2,c1);
endmodule
```

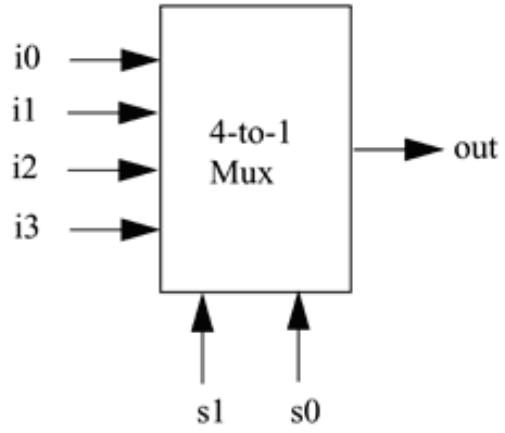


Bài 3: Bộ cộng đầy đủ 1 bit và 4 bit

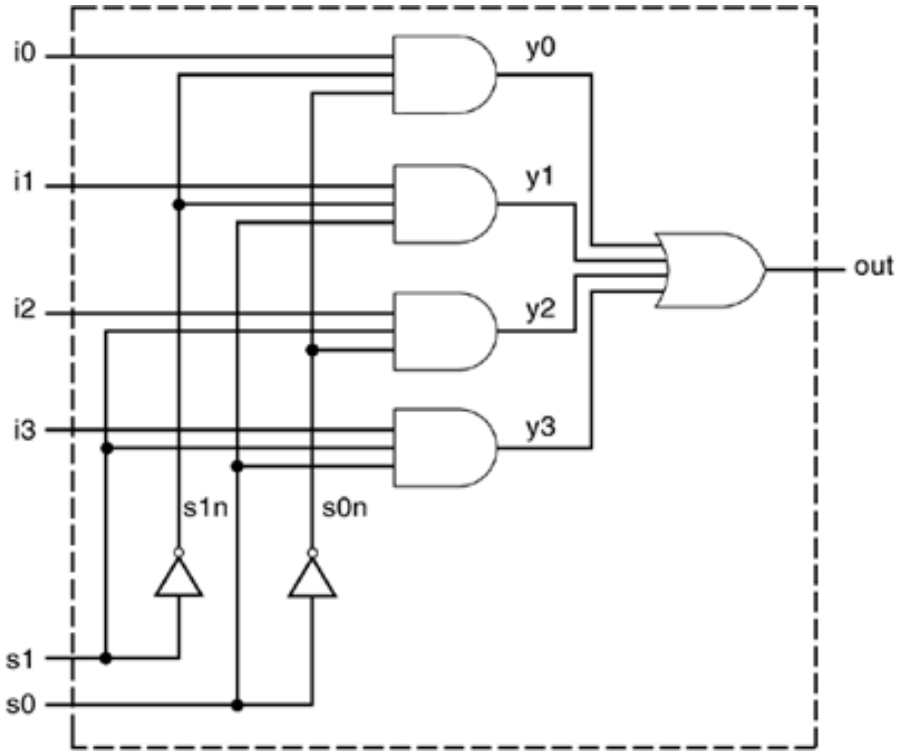
```
module fulladder4 (sum, c_out, a, b, c_in);
//khai báo I/O port


output[3:0] sum;
output c_out;
input [3:0] a,b;
input c_in;
wire a1, c2, c3;


//ghép 4 bộ cộng 1 bit
fulladder fa0(sum[0], c1, a[0], b[0],c_in);
fulladder fa1(sum[1], c2, a[1], b[1],c1);
fulladder fa2(sum[2], c3, a[2], b[2],c2);
fulladder fa3(sum[3], c_out, a[3], b[3],c3);
endmodule
```



Bài 4: Bộ ghép kênh 4:1



s1	s0	out
0	0	I0
0	1	I1
1	0	I2
1	1	I3



Bài 4: Bộ ghép kênh 4:1

```
module mux4_to_1 (out, i0, i1, i2, i3, s1, s0);

output out;
input i0, i1, i2, i3;
input s1,s0;
wire s1n,s0n;
wire y0,y1, y2, y3;

not (s1n,s1);
not (s0n;s0);
and (y0, i0, s1n, s0n);
and (y1, i1, s1n, s0);
and (y2, i2, s1, s0n);
and (y3, i3, s1, s0);
or (out, y0, y1, y2, y3);
endmodule
```

Bài 4: Bộ ghép kênh 4:1

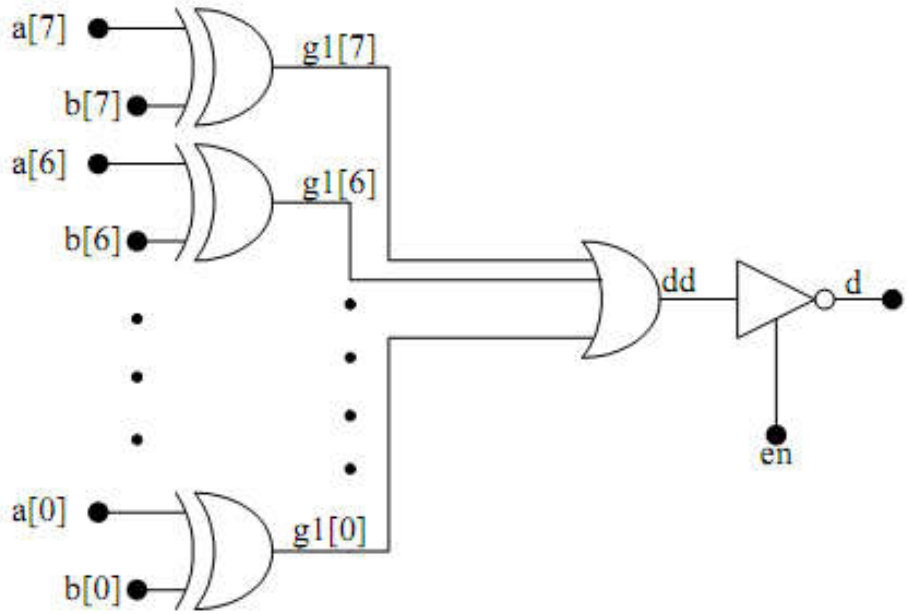
```
1  module mux_from_gates ();
2  reg c0,c1,c2,c3,A,B;
3  wire Y;
4  //Invert the sel signals
5  not (a_inv, A);
6  not (b_inv, B);
7  // 3-input AND gate
8  and (y0,c0,a_inv,b_inv);
9  and (y1,c1,a_inv,B);
10 and (y2,c2,A,b_inv);
11 and (y3,c3,A,B);
12 // 4-input OR gate
13 or (Y, y0,y1,y2,y3);
14
15 // Testbench Code goes here
16 initial begin
17    $monitor (
18     "c0 = %b c1 = %b c2 = %b c3 = %b A = %b B = %b Y = %b",
19     c0, c1, c2, c3, A, B, Y);
```

```
c0 = 0 c1 = 0 c2 = 0 c3 = 0 A = 0 B = 0 Y = 0
c0 = 1 c1 = 0 c2 = 0 c3 = 0 A = 1 B = 0 Y = 0
c0 = 0 c1 = 1 c2 = 0 c3 = 0 A = 1 B = 0 Y = 0
c0 = 1 c1 = 1 c2 = 1 c3 = 0 A = 1 B = 1 Y = 0
c0 = 0 c1 = 0 c2 = 1 c3 = 1 A = 1 B = 1 Y = 1
c0 = 1 c1 = 0 c2 = 1 c3 = 1 A = 1 B = 1 Y = 1
c0 = 0 c1 = 1 c2 = 0 c3 = 1 A = 1 B = 1 Y = 1
c0 = 1 c1 = 1 c2 = 0 c3 = 1 A = 0 B = 1 Y = 1
c0 = 0 c1 = 0 c2 = 0 c3 = 0 A = 0 B = 1 Y = 0
c0 = 1 c1 = 0 c2 = 1 c3 = 0 A = 0 B = 1 Y = 0
c0 = 0 c1 = 1 c2 = 1 c3 = 0 A = 0 B = 1 Y = 1
c0 = 1 c1 = 1 c2 = 1 c3 = 0 A = 0 B = 1 Y = 1
c0 = 0 c1 = 0 c2 = 0 c3 = 1 A = 0 B = 1 Y = 0
c0 = 1 c1 = 0 c2 = 0 c3 = 1 A = 0 B = 1 Y = 0
c0 = 0 c1 = 1 c2 = 0 c3 = 1 A = 0 B = 1 Y = 1
```

```
20    c0 = 0;
21    c1 = 0;
22    c2 = 0;
23    c3 = 0;
24    A = 0;
25    B = 0;
26    #1  A  = 1;
27    #2  B  = 1;
28    #4  A  = 0;
29    #8  $finish;
30 end
31
32 always #1  c0 = ~c0;
33 always #2  c1 = ~c1;
34 always #3  c2 = ~c2;
35 always #4  c3 = ~c3;
36
37 endmodule
```

Bài 5: Bộ so sánh 8 bit

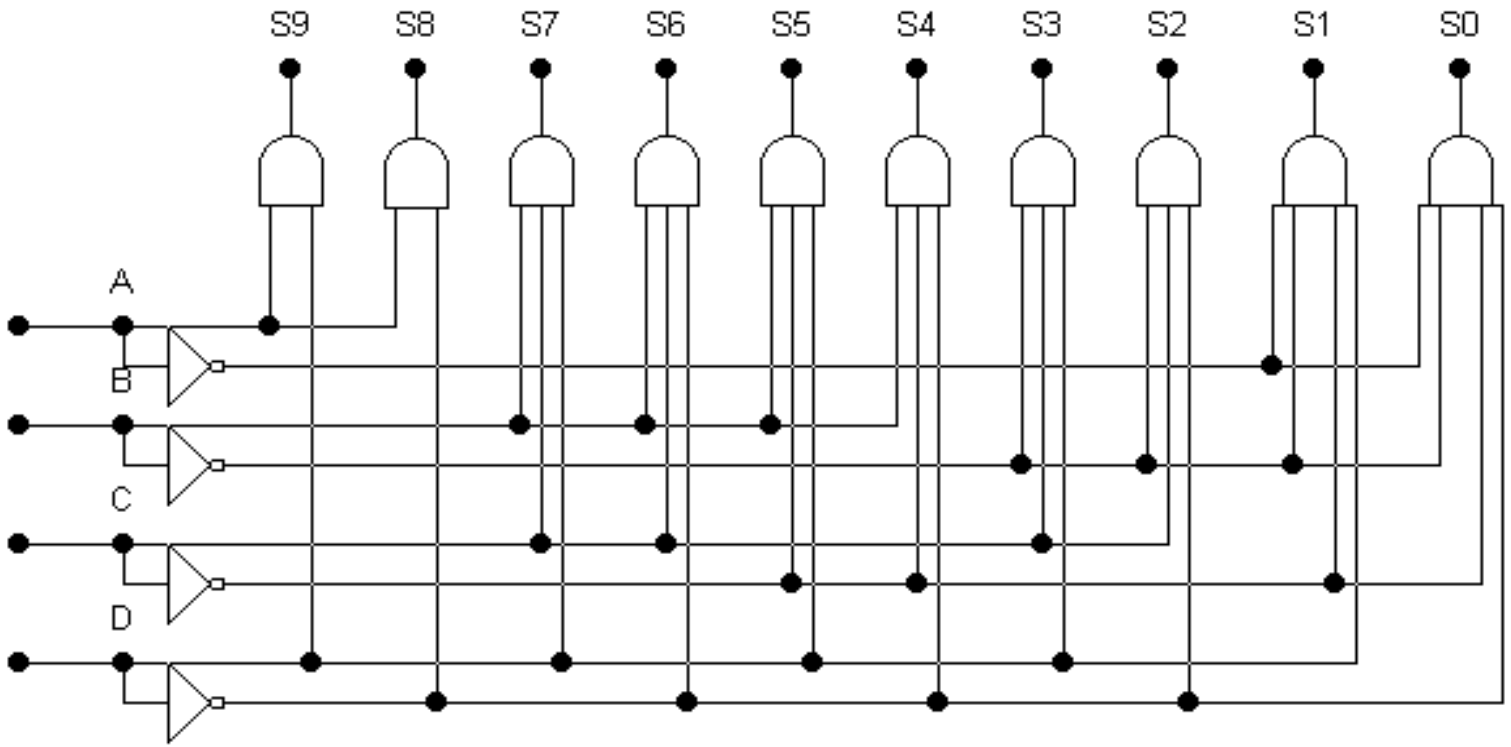
- Viết module Verilog và testbench mô phỏng mô tả bộ so sánh 2 giá trị byte *a* và *b* theo mô hình mức cổng
  - Giá trị đầu ra d sẽ =1 nếu a=b, =0 trong các trường hợp còn lại.
  - Đầu ra được kích hoạt nếu en=1; nếu en=0 thì d=z.



Bài 5: Bộ so sánh 8 bit

```
module comp(d,a,b,en);
input en;
input[7:0]a,b;
output d;
wire [7:0]c;
wire dd;
xor g1[7:0](c,b,a);
or(dd,c);
notif1(d,dd,en);
Endmodule
//============================
module comp_tb;
reg[7:0]a,b;
reg en;
comp gg(d,a,b,en);
initial
begin
a  = 8'h00;
b  = 8'h00;
en = 1'b0;
end
always
#2  en = 1'b1;
always
begin
 #2  a = a+1'b1;
 #2  b = b+2'd2;
end
initial $monitor($time," en = %b , a
= %b ,b = %b ,d =
%b ",en,a,b,d);
initial #30 $stop;
endmodule
```

Bài 6: Bộ mã hoá BCD



Bài 6: Bộ mã hoá BCD

Chữ số thập phân	Từ mã nhị phân
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001