数字系统设计课堂练习（1）2020_11_3 参考答案

一、基本语法

（1）请根据以下两条语句，从选项中找出正确的答案。

①
```
reg [7:0] A;  reg [1:0] B;
A = 8'hFF;    B = A;    A = B;
```

A = a) 8'b0000_0011 ✓ b) 7'h03 c) 8'b1111_1111 d) 8'd1515

②
```
reg [7:0] B, C;    B = 8'bZ0;    C = 6'b0x1;
```

B = a) 8'b0000_00Z0 b) 8'bZZZZ_0000 c) 8'b0000_ZZZ0 d) 8'bZZZZ_ZZZ0 ✓

C= a) 8'b0XXX_XXX1 b) 8'bXXXX_XXX1 c) 8'b0000_00X1 ✓ d) 8'b000X_XXX1

如果某数的最高位为 0，x 或 z，则用 0，x 或 z 填满余下的最高位。

③
```
reg [11:0] Data;
Data = 11'bx10_10Z0;
```

Data = a) 12'b0XXX_XX10_10Z0 ✓ b) 11'bXXX_XX10_10Z0

c) 12'b0000_0x10_10Z0 d) 6'b10_10Z0

④
```
reg [7:0] num1,  num2;
reg [8:0] f;
num1 = -8'd26;   num2 = -8'd10;
f = num1 + num2;
```

num1 = 8'b1110_0110 (-8'd26 补码)

num2 = 8'b1111_0110 (-8'd10 补码)

f = 9'b1_1101_1100

f = a) 8'b1101_1100 b) -9'd36

c) 9'b1_1101_1100 ✓ d) -8'd36

（2）根据下面的语句片段，判断其中的变量：cout, cin, psum, pb 的类型。并说明理由。

```
modul fadder( cout, sum, a, b, cin);
   input   a, b, cin;
   output cout, sum;
   ……
endmodule

modul testbench( );
   ……
   fadder  m_u( c1, psum, pa, pb, c0);
   ……
endmodule
```

答:cout 可以是 reg/wire，cin 只能是 wire,psum 只能是 wire，pb 可以是 reg/wire。

（3）下面是一个 Verilog 模块中的部分语句：

```
reg  [6:0]  a = 5'b0_1111;
reg  [6:0]  b =  4'b1101;
reg  [7:0]  F;
reg  [5:0]  G;
F = a + b;
G = a ^ b;
```

根据这段程序，确定运行后的结果：G= 6'b00_0010 ，F= 8'b0001_1100 。

（4）下面是一个 Verilog 模块中的部分语句：

```
reg  [4:0]  a = 5'b0_1011;
reg  [4:0]  b = 5'b1_0Z10;
reg  [5:0]  F1;
reg  [7:0]  F2;
reg  [3:0]  F3
F1 = a + b;
F2 = a ^ b;
F3 = ( F1 ) ? a : F2;
```

根据这段程序，确定运行后的结果：

F1 = 6'bxx_xxxx ， F2 = 8'b0001_1x01 ， F3 = 4'b1xx1

（5）**只使用门原语语句，不使用其它任何语句**，重新改写下面的 Verilog 模块。

```
module comb(output y,  input i0, i1, s);
    wire #(0.6, 0.8)
        im0 = i0 & ~s,
        im1 = i1 & s;
    assign #(3, 4) y = im0 | im1;
endmodule
```

答：使用门原语

```
module comb_gate(output y,  input i0, i1, s);
    wire sbar, im0, im1;
    not u0(sbar, s );
    and #(0.6, 0.8)
         u10(im0, i0, sbar),
         u11(im1, i1, s   );
    or #(3, 4) u20(y, im0, im1);
endmodule
```

（6）① 找出下面模块中包含错误，将其更改，并说明理由。

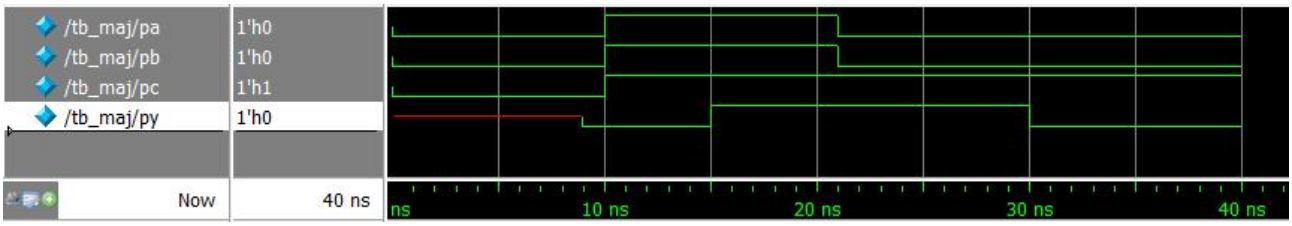
```
module maj ( output y, input a, b, c );

    reg [3:1] p;  // 错误，应该为： wire [3:1] p; 门原语只能驱动线网

    and #(2, 4)
        u10( p[1], a, b ),
        u11( p[2], b, c ),
        u12( p[3], c, a );
    or #(3, 5) (y, p[1], p[2], p[3] );
endmodule
```

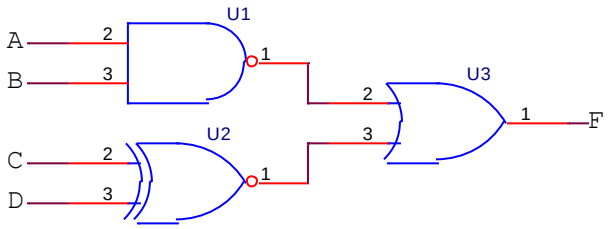
② 在 t = 0 时刻： a = 1'b0, b = 1'b0, c = 1'b0;
在 t = 10 时刻： a = 1'b1, b = 1'b1, c = 1'b1;
在 t = 21 时刻： a = 1'b0, b = 1'b0, c = 1'b1;
在 t = **30** 时， y = 1'b0

答：and 门的下降延时加 or 门的下降延时 21 + 4 + 5



二、简单模块

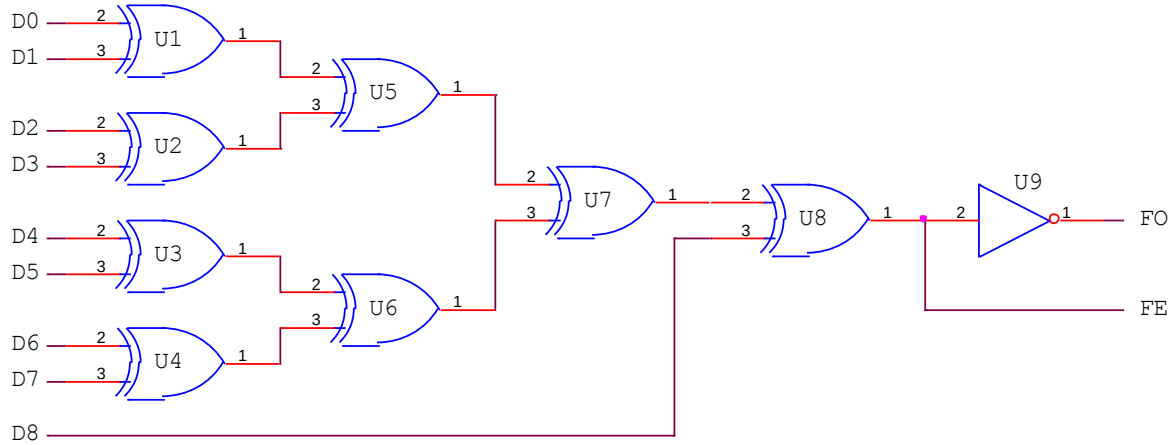
（1）根据下面的电路原理图，使用门原语设计 Verilog 模块；模块名为 comb；




```
// File: comb.v
module comb( output F, input A, B, C, D);
    wire w1, w2;

   nand u1 ( w1, A, B );
   xnor u2 ( w2, C, D );
   or   u3 ( F, w1, w2);
endmodule
```

（2）根据下面的电路原理图，使用门原语设计 Verilog 模块；模块名为：parity9。




```
// File: parity.v

module parity9( output FO, FE, input [8:0] D );

    wire w11, w12, w13, w14;
   wire w21, w22;
   wire w3, w4;

   xor u1( w11, D[1], D[0]),
       u2( w12, D[3], D[2]),
      u3( w13, D[5], D[4]),
      u4( w14, D[7], D[6]),
      //
      u5( w21, w12, w11),
      u6( w22, w14, w13),
      //
      u7( w3, w22, w21),
      //
      u8( w4, w3, D[8]);

   not u9( FO, w4);
   buf u10( FE, w4);
endmodule
```

（3）根据下面的电路原理图，使用门原语设计 Verilog 模块，其中，图 3-3a 中的模块 2x4 decoder 的原理图如图 3-3b 所示；首先，设计图 3-3b 中的模块，模块名为 decoder2x4，然后通过调用 decoder2x4 设计图 3-3a 中的模块，模块名为 mux4x1。

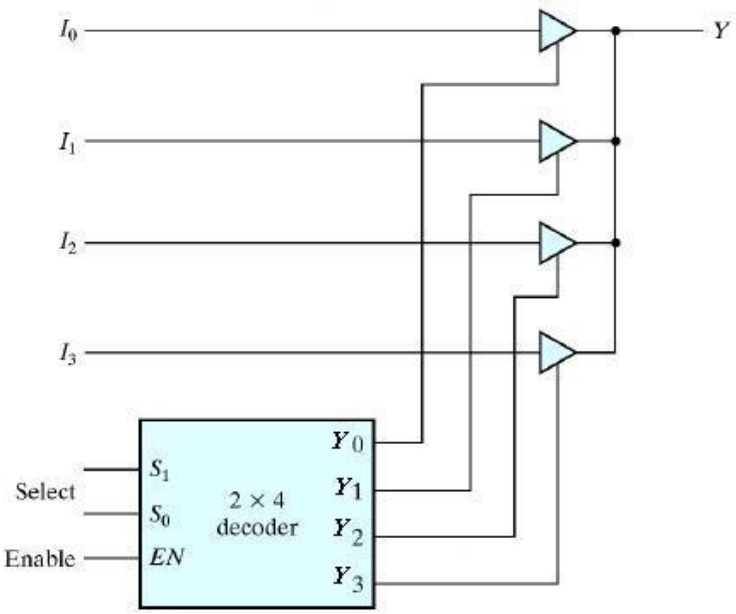


图 3-3a、4 选一多路选择器

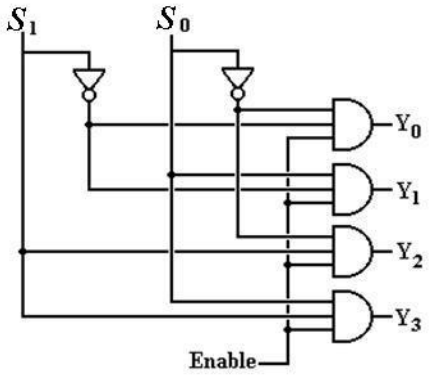


图 3-3b、图 3-3a 中的 2x4 decoder

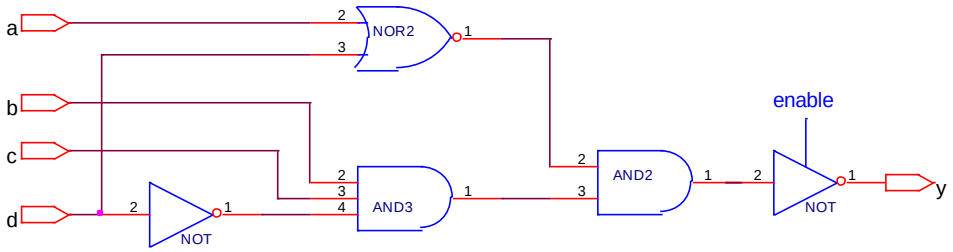
```
// File: decoder2x4.v
module decoder2x4( output [3:0] Y, input S1, S0, Enable );
    wire wn1, wn0;

    not u10(wn1, S1),
        u11(wn0, S0);
    and u23( Y[3], S1, S0, Enable),
        u22( Y[2], S1, wn0, Enable),
        u21( Y[1], wn1, S0, Enable),
        u20( Y[0], wn1, wn0, Enable);
endmodule
```

```
// File: mux4x1.v
`include "decoder2x4.v"
module mux4x1( output Y, input [3:0] I, input [1:0] select, input Enable );
    wire [3:0] pY;

    decoder2x4 m1( .Y(pY), .S1(select[1]), .S0(select[0]), .Enable(Enable) );
    bufif1 u0( Y, I[0], pY[0]),
           u1( Y, I[1], pY[1]),
           u2( Y, I[2], pY[2]),
           u3( Y, I[3], pY[3]);
endmodule
```

（4）根据下面的电路原理图，**只使用连续赋值语句**，完成相应的 Verilog 模块设计。




```
module comb4(output y, input a, b, c, d, enable);

    wire w10, w20, w21, w30;

    assign w10 = ~d,
           w20 = ~(a | d),
           w21 = b & c & w10,
           w30 = w20 & w21,
           y = (enable) ? ~(w30) : 1'bz;

endmodule
```