IC设计-第一次作业

刘沁雨 2021211039

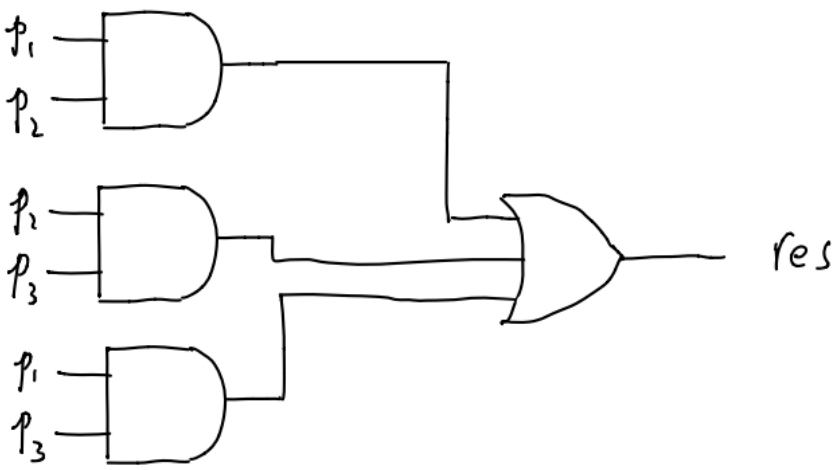
1.

RTL代码

```
module hw_1_1 (
    input  p1,p2,p3,
    output res
);
assign res = (p1&p2)|(p2&p3)|(p1&p3);

endmodule
```

原理图



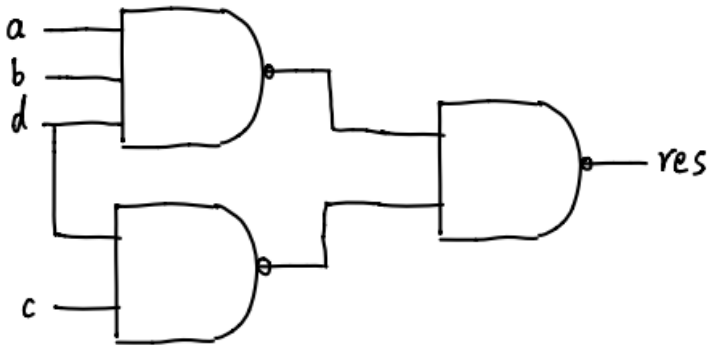
2.

RTL代码

```
module hw_1_2(
    input a,b,c,d,
    output res
);
wire res_1,res_2;
assign res_1 = ~(a&b&d);
assign res_2 = ~(c&d);
assign res = ~(res_1&res_2);

endmodule
```

原理图

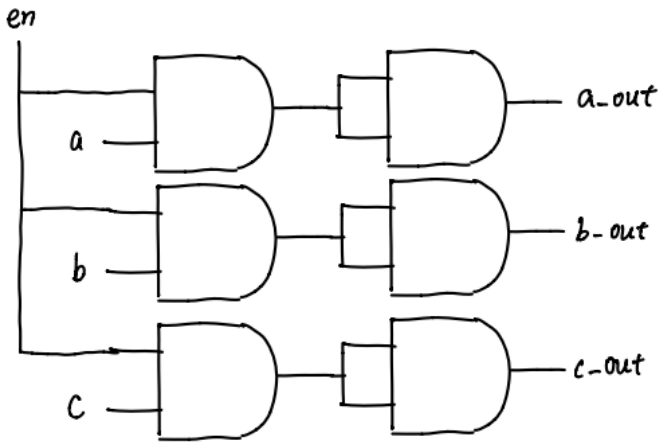


3.

RTL代码

```
module hw_1_3(
     input      [2:0]abc,en,
     output reg [2:0]abc_out
);
always@(abc or en)
    if(!en)
        abc_out = 3'b0;
    else
        abc_out = abc;
endmodule
```

原理图



4.

RTL代码

```
module hw_1_4 (
    input       [15:0]temp_code,
    output  reg  [7:0]bcd_code
```

```
);

always@(temp_code)
begin
    case(temp_code)
        16'b0000_0000_0000_0000: bcd_code = 8'b0000_0000;
        16'b0000_0000_0000_0001: bcd_code = 8'b0000_0001;
        16'b0000_0000_0000_0011: bcd_code = 8'b0000_0010;
        16'b0000_0000_0000_0111: bcd_code = 8'b0000_0011;//3

        16'b0000_0000_0000_1111: bcd_code = 8'b0000_0100;
        16'b0000_0000_0001_1111: bcd_code = 8'b0000_0101;
        16'b0000_0000_0011_1111: bcd_code = 8'b0000_0110;
        16'b0000_0000_0111_1111: bcd_code = 8'b0000_0111;//7

        16'b0000_0000_1111_1111: bcd_code = 8'b0000_1000;
        16'b0000_0001_1111_1111: bcd_code = 8'b0000_1001;
        16'b0000_0011_1111_1111: bcd_code = 8'b0001_0000;
        16'b0000_0111_1111_1111: bcd_code = 8'b0001_0001;//11

        16'b0000_1111_1111_1111: bcd_code = 8'b0001_0010;
        16'b0001_1111_1111_1111: bcd_code = 8'b0001_0011;
        16'b0011_1111_1111_1111: bcd_code = 8'b0001_0100;
        16'b0111_1111_1111_1111: bcd_code = 8'b0001_0101;//15

        16'b1111_1111_1111_1111: bcd_code = 8'b0001_0110;//16
        default: bcd_code = 8'b1111_1111;
    endcase
end
endmodule
```

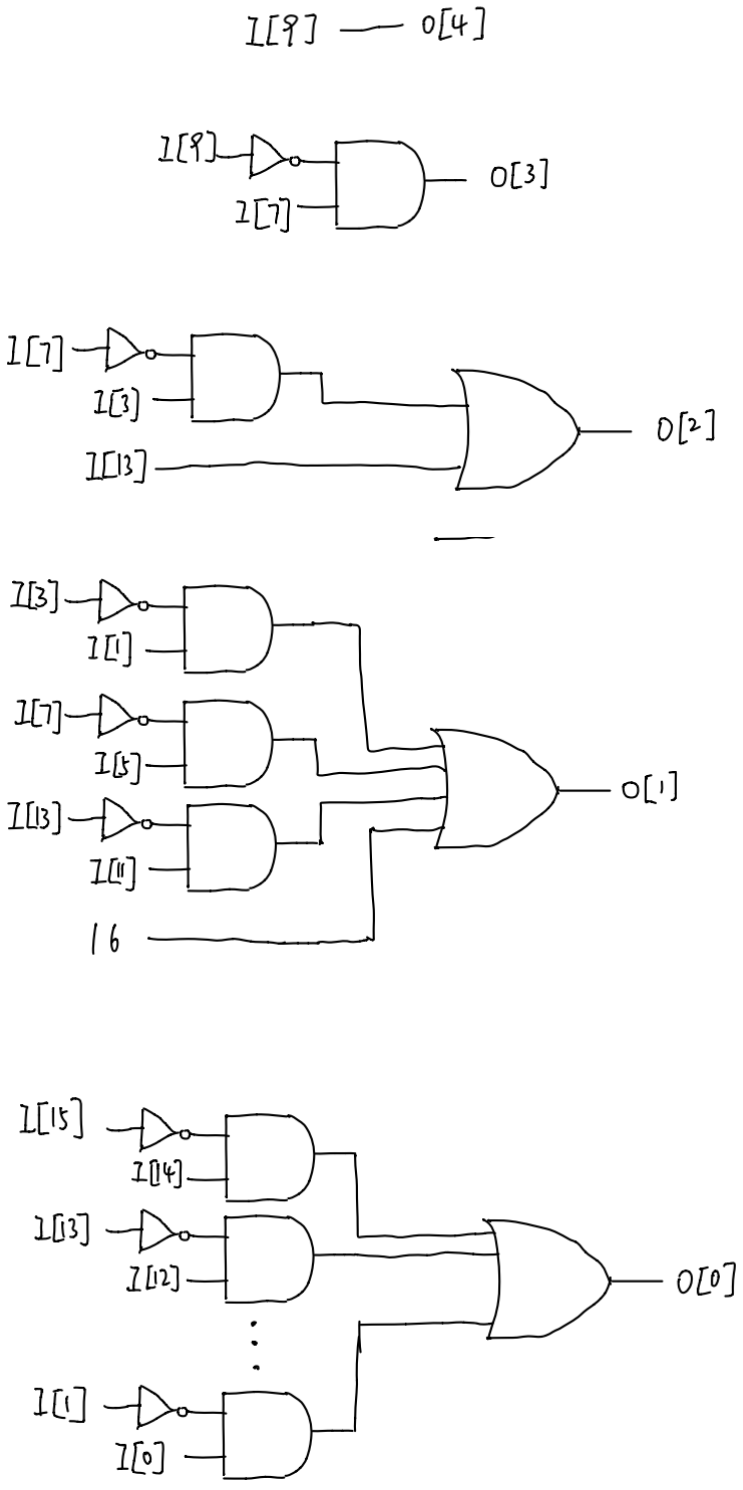
直接16位转8位的编码器比较好写，但不好画出原理图，于是重新想了个新的用门电路搭的逻辑。

因为热码的空状态比较多，比如前三位一直保持为0，刚好可以去掉很多无关项，仅适用此题，没有拓展性。

RTL代码

```
module hw_1_4_2 (
    input       [15:0]I,
    output       [7:0]O
);
    assign O[7] = 0;
    assign O[6] = 0;
    assign O[5] = 0;
    assign O[4] = I[9];
    assign O[3] = ~I[9]&I[7];
    assign O[2] = ( ~I[7]&I[3] | I[13] );
    assign O[1] = ( (~I[3]&I[1]) | (~I[7]&I[5]) | ~(I[13]&I[11]) | I[16] );
    assign O[0] = ((~I[15]&I[14])|(~I[13]&I[12])| (~I[11]&I[10])|
(~I[9]&I[8])
                  | (~I[7]&I[6]) | (~I[5]&I[4]) | (~I[3]&I[2]) |
(~I[1]&I[0]) );
endmodule
```

原理图


I[9]
O[4]
I[9]
I[7]
O[3]
I[7]
I[3]
I[13]
O[2]
I[3]
I[1]
I[7]
I[5]
I[13]
I[11]
16
O[1]
I[15]
I[14]
I[3]
I[12]
I[1]
I[0]
O[0]

5.

RTL代码

```
module hw_1_5 (
    input       [15:0]true_code,
    output  reg [15:0]comp_code
);
    always@(true_code)
    begin
        if(true_code > 16'b0111_1111_1111_1111)
            comp_code = ~true_code + 16'b0000_0000_0000_0001;
        else
            comp_code = true_code;
    end
endmodule
```

原理图