1.8-3编码器

```
module bianma8_3(i,y);
 input [7:0] i;
 output [2:0] y;
 reg [2:0] y;
 always @(i)
 begin
   case (i[7:0])
   8'b0000_0001: y[2:0]=3'b000;
   8'b0000_0010: y[2:0]=3'b001;
   8'b0000_0100: y[2:0]=3'b010;
   8'b0000_1000: y[2:0]=3'b011;
   8'b0001_0000: y[2:0]=3'b100;
   8'b0010_0000: y[2:0]=3'b101;
   8'b0100_0000: y[2:0]=3'b110;
   8'b1000_0000: y[2:0]=3'b111;
   default: y[2:0]=3'b000;
   endcase
 end
endmodule
```

2.3-8译码器

```
module decoder3_8(a,g1,g2,g3,y);
input [2:0] a;
 output [7:0] y;
 input g1,g2,g3;
 reg [7:0] y;
 always @(a,g1,g2,g3)
  begin
  if(g1==0) y=8'b1111_1111;
  else if (g2==1) y=8'b1111_1111;
  else if (g3==1) y=8'b1111_1111;
  else
    case (a[2:0])
    3'b000: y[7:0]=8'b1111_1110;
    3'b001: y[7:0]=8'b1111_1101;
    3'b010: y[7:0]=8'b1111_1011;
    3'b011: y[7:0]=8'b1111_0111;
    3'b100: y[7:0]=8'b1110_1111;
    3'b101: y[7:0]=8'b1101_1111;
    3'b110: y[7:0]=8'b1011_1111;
    3'b111: y[7:0]=8'b0111_1111;
    endcase
  end
 endmodule
```

3. 四选一数据选择器.

```
module mux4-1(a, y, d0, d1, d2, d3, g);
 input [1:0] a;
 output y;
 input d0, d1, d2, d3;
 input g;
 reg y;
 always@(a, d0, d1, d2, d3, g)
  begin
   if(g==0)
     y=0;
   else
     case(a[1:0])
    2'b00:  y= d0;
    2'b01:  y= d1;
    2'b10:  y= d2;
    2'b11:  y= d3;
    endcase
  end
endmodule.
```

4. 数据分配器

```
module dmux4-1(a, y0, y1, y2, y3, din);
 input [1:0] a;
 input din;
 output y0, y1, y2, y3;
 reg y0, y1, y2, y3;
always @(a, din)
begin
   y0=0; y2=0; y3=0; y1=0;
   case(a[1:0])
   2'b00: y0= din;
   2'b01: y1= din;
   2'b10: y2= din;
   2'b11: y3= din;
   endcase
 end
endmodule.
```

5. 数值比较器.

```
module comparator (a, b, y1, y2, y3);
 input [3:0] a, b;
  output y1, y2, y3;
  reg y1, y2. y3;
  always @(a, b)
   begin
     if (a>b)
      y1 = 1; y2 = 0; y3 = 0;
     else if (a = = b)
      y1 = 0; y2 = 1; y3 = 0;
     else if (a<b)
      y1 = 0; y2 = 0; y3 = 1;
      end
endmodule.
```

6 半加.

```
module half_add(a,b,sum,cout);
 input a,b;
 output sum,cout;
 reg sum,cout;
  always@(a,b)
  begin
   sum= a^b;
   cout= a&b;
   end
endmodule
```

全加.

```
module add(a,b,cin,cout,sum);
input a,b,cin;
output sum,cout;
reg sum,cout;
always@(a,b,cin)
begin
 sum=(a^b)^c
 cout=(a&b)|(a&cin)|(b&cin);
end
endmodule
```

半减.

```
module half_sub(a,b,dout,cout);
 input a,b;
 output dout,cout;
 reg dout,cout;
  always@(a,b)
  begin
   {cout,dout}= a-b;
   end
endmodule.
```

全减.

```
module sub(a,b,cin,dout,cout);
input a,b,cin;
output dout,cout;
reg dout,cout;
always @(a,b,cin)
begin
 {cout,dout}= a-b-cin;
end
endmodule.
```

7. 异步复位.

```
module flipflop(D. Q. clk. rst);
 input D, clk,rst;
 output Q;
 reg Q;
 always@(posedge clk or negedge rst)
  begin
    if(!rst)
      Q<=0;
    else
      Q<=D;
  end
endmodule
```

```
module flipflop(j,k, q, nq, clk. rst);
 input j,k. clk. rst;
 output q. nq;
 reg q
 assign nq=~q;
 always @(posedge clk or negedge rst)
 begin
   if(!rst)
   q<= 1'b0;
   else
    case({j,k})
    2'b00: q<=q;
    2'b01: q<=1'b0;
    2'b10: q<=1'b1;
    2'b11: q<=nq;
    endcase
  end
endmodule.
```