七段数码管动态显示电路

徐震 PB20051102

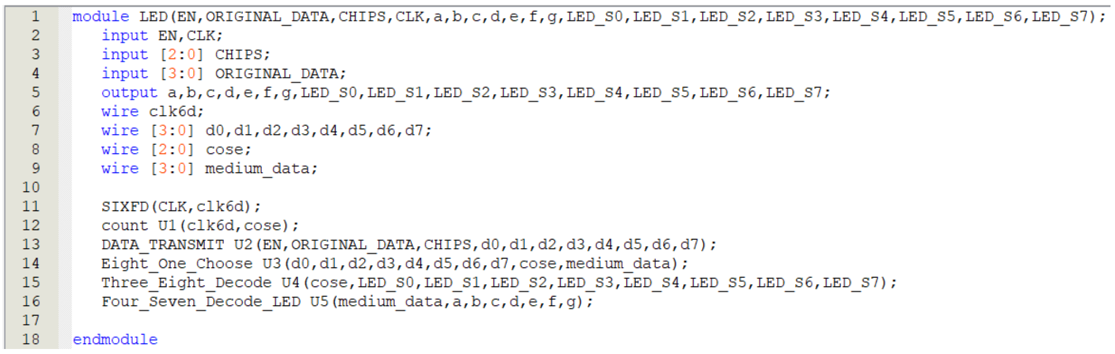
实验内容

通过时钟生成位选信号选择位置；通过拨码开关控制段选信号经 4-7 译码器翻译之后使 LED 数码管显示出想要的数字。

设计分析

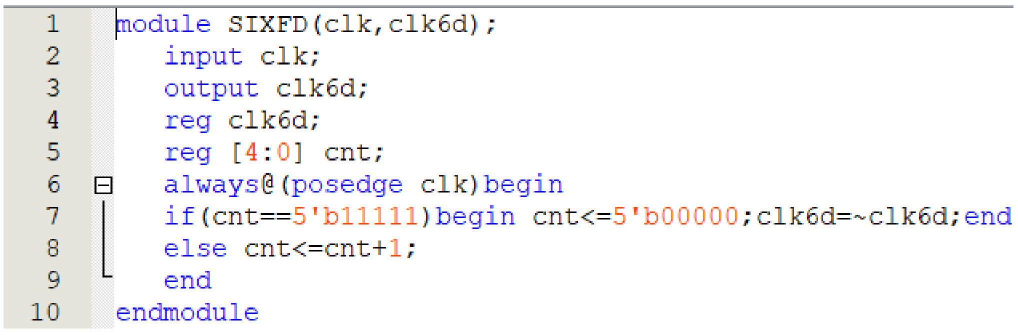
先经过锁存器使得段选信号可以被保存，通过八进制计数器生成位选信号选择被改变的锁存器，将计数器三八译码得到最终的位选信号，同时将锁存器的输出四七译码生成 LED 数码管显示信号。

源码



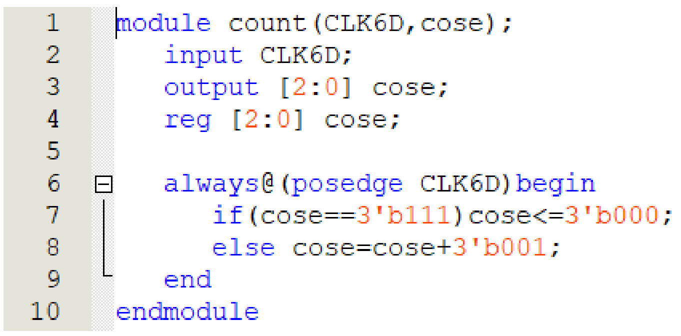
```verilog
module LED(EN,ORIGINAL_DATA,CHIPS,CLK,a,b,c,d,e,f,g,LED_S0,LED_S1,LED_S2,LED_S3,LED_S4,LED_S5,LED_S6,LED_S7);
   input EN,CLK;
   input [2:0] CHIPS;
   input [3:0] ORIGINAL_DATA;
   output a,b,c,d,e,f,g,LED_S0,LED_S1,LED_S2,LED_S3,LED_S4,LED_S5,LED_S6,LED_S7;
   wire clk6d;
   wire [3:0] d0,d1,d2,d3,d4,d5,d6,d7;
   wire [2:0] cose;
   wire [3:0] medium_data;

   SIXFD(CLK,clk6d);
   count U1(clk6d,cose);
   DATA_TRANSMIT U2(EN,ORIGINAL_DATA,CHIPS,d0,d1,d2,d3,d4,d5,d6,d7);
   Eight_One_Choose U3(d0,d1,d2,d3,d4,d5,d6,d7,cose,medium_data);
   Three_Eight_Decode U4(cose,LED_S0,LED_S1,LED_S2,LED_S3,LED_S4,LED_S5,LED_S6,LED_S7);
   Four_Seven_Decode_LED U5(medium_data,a,b,c,d,e,f,g);

endmodule
```

顶层实体



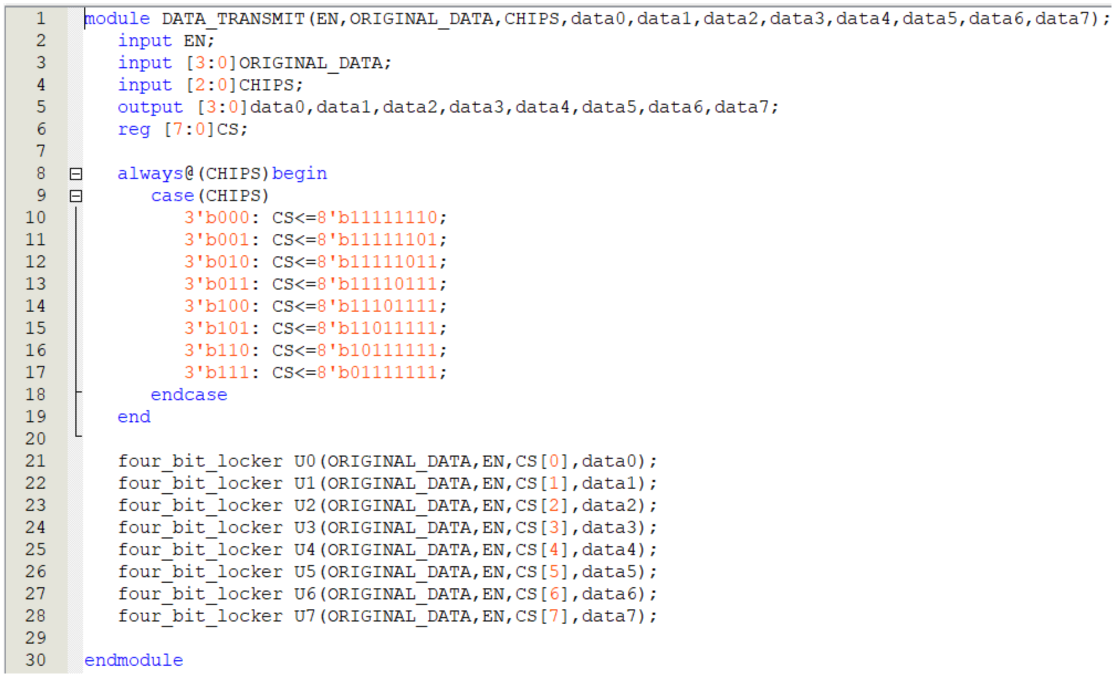
```verilog
module SIXFD(clk,clk6d);
   input clk;
   output clk6d;
   reg clk6d;
   reg [4:0] cnt;
   always@(posedge clk)begin
   if(cnt==5'b11111)begin cnt<=5'b00000;clk6d=~clk6d;end
   else cnt<=cnt+1;
   end
endmodule
```

六分频器



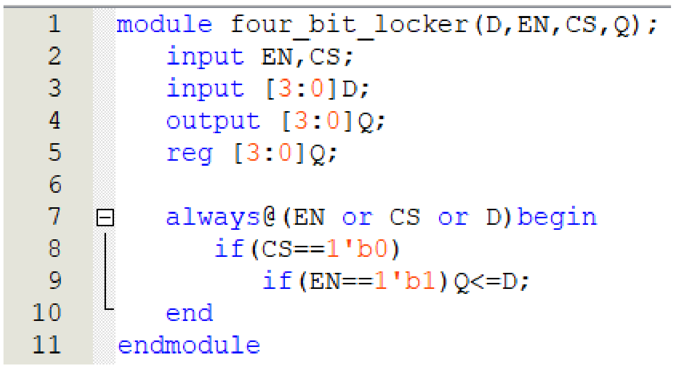
```verilog
module count(CLK6D,cose);
    input CLK6D;
    output [2:0] cose;
    reg [2:0] cose;

    always@(posedge CLK6D)begin
        if(cose==3'b111)cose<=3'b000;
        else cose=cose+3'b001;
    end
endmodule
```

计数器



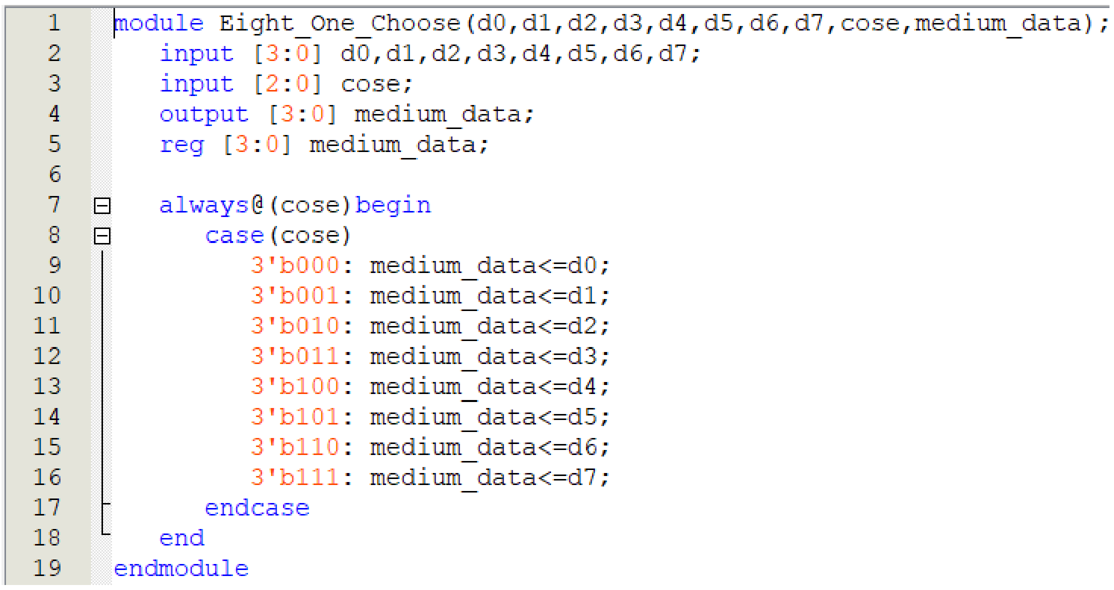
```verilog
module DATA_TRANSMIT(EN,ORIGINAL_DATA,CHIPS,data0,data1,data2,data3,data4,data5,data6,data7);
    input EN;
    input [3:0]ORIGINAL_DATA;
    input [2:0]CHIPS;
    output [3:0]data0,data1,data2,data3,data4,data5,data6,data7;
    reg [7:0]CS;

    always@(CHIPS)begin
        case(CHIPS)
            3'b000: CS<=8'b11111110;
            3'b001: CS<=8'b11111101;
            3'b010: CS<=8'b11111011;
            3'b011: CS<=8'b11110111;
            3'b100: CS<=8'b11101111;
            3'b101: CS<=8'b11011111;
            3'b110: CS<=8'b10111111;
            3'b111: CS<=8'b01111111;
        endcase
    end

    four_bit_locker U0(ORIGINAL_DATA,EN,CS[0],data0);
    four_bit_locker U1(ORIGINAL_DATA,EN,CS[1],data1);
    four_bit_locker U2(ORIGINAL_DATA,EN,CS[2],data2);
    four_bit_locker U3(ORIGINAL_DATA,EN,CS[3],data3);
    four_bit_locker U4(ORIGINAL_DATA,EN,CS[4],data4);
    four_bit_locker U5(ORIGINAL_DATA,EN,CS[5],data5);
    four_bit_locker U6(ORIGINAL_DATA,EN,CS[6],data6);
    four_bit_locker U7(ORIGINAL_DATA,EN,CS[7],data7);

endmodule
```

八个四比特锁存器



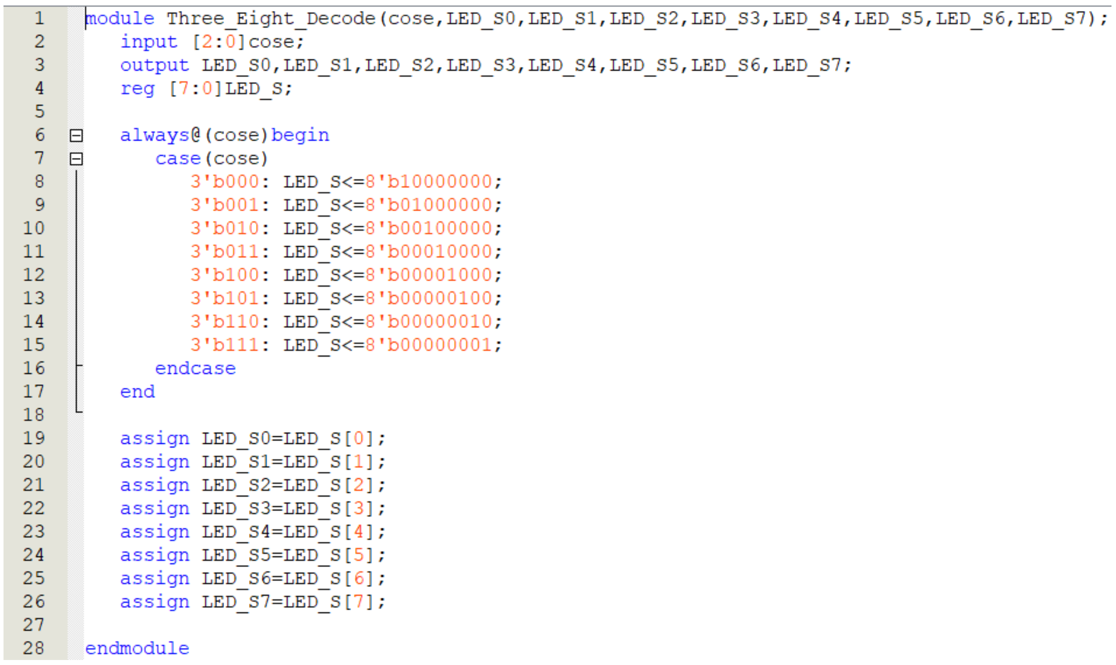
```verilog
module four_bit_locker(D,EN,CS,Q);
    input EN,CS;
    input [3:0]D;
    output [3:0]Q;
    reg [3:0]Q;

    always@(EN or CS or D)begin
        if(CS==1'b0)
            if(EN==1'b1)Q<=D;
    end
endmodule
```

四比特锁存器



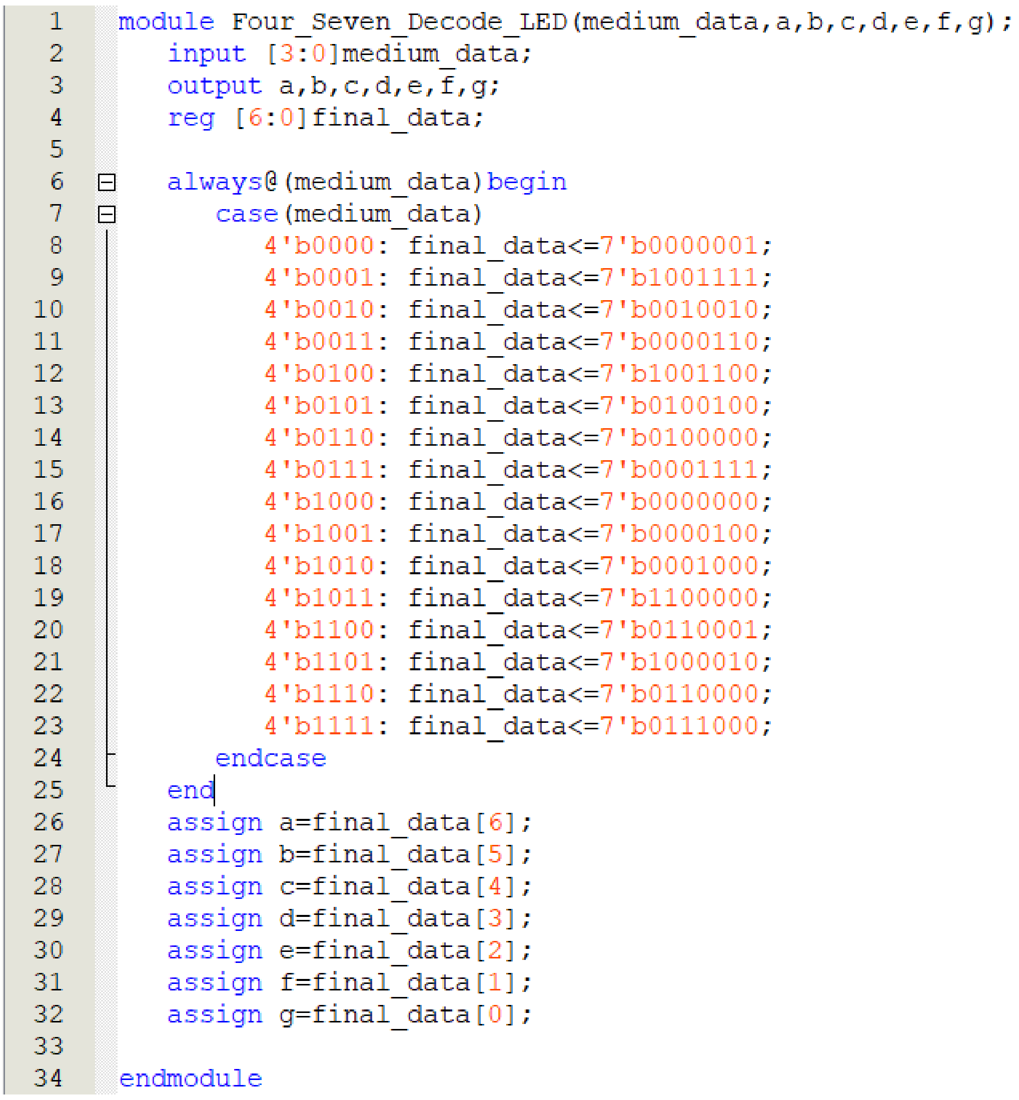
```verilog
module Eight_One_Choose(d0,d1,d2,d3,d4,d5,d6,d7,cose,medium_data);
   input [3:0] d0,d1,d2,d3,d4,d5,d6,d7;
   input [2:0] cose;
   output [3:0] medium_data;
   reg [3:0] medium_data;

   always@(cose)begin
      case(cose)
         3'b000: medium_data<=d0;
         3'b001: medium_data<=d1;
         3'b010: medium_data<=d2;
         3'b011: medium_data<=d3;
         3'b100: medium_data<=d4;
         3'b101: medium_data<=d5;
         3'b110: medium_data<=d6;
         3'b111: medium_data<=d7;
      endcase
   end
endmodule
```

八选一选择器



```verilog
module Three_Eight_Decode(cose,LED_S0,LED_S1,LED_S2,LED_S3,LED_S4,LED_S5,LED_S6,LED_S7);
   input [2:0]cose;
   output LED_S0,LED_S1,LED_S2,LED_S3,LED_S4,LED_S5,LED_S6,LED_S7;
   reg [7:0]LED_S;

   always@(cose)begin
      case(cose)
         3'b000: LED_S<=8'b10000000;
         3'b001: LED_S<=8'b01000000;
         3'b010: LED_S<=8'b00100000;
         3'b011: LED_S<=8'b00010000;
         3'b100: LED_S<=8'b00001000;
         3'b101: LED_S<=8'b00000100;
         3'b110: LED_S<=8'b00000010;
         3'b111: LED_S<=8'b00000001;
      endcase
   end

   assign LED_S0=LED_S[0];
   assign LED_S1=LED_S[1];
   assign LED_S2=LED_S[2];
   assign LED_S3=LED_S[3];
   assign LED_S4=LED_S[4];
   assign LED_S5=LED_S[5];
   assign LED_S6=LED_S[6];
   assign LED_S7=LED_S[7];

endmodule
```

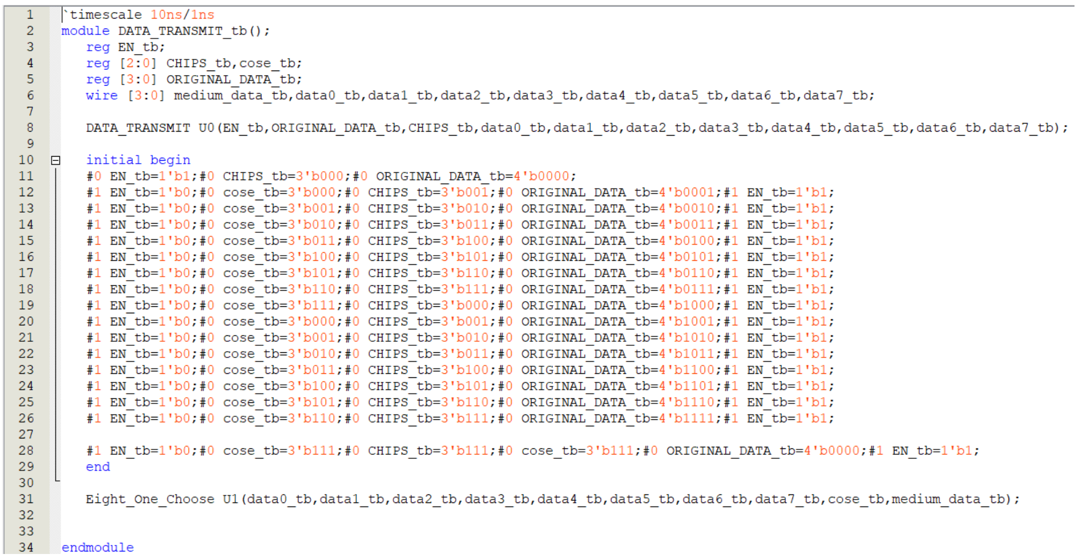
三八译码器



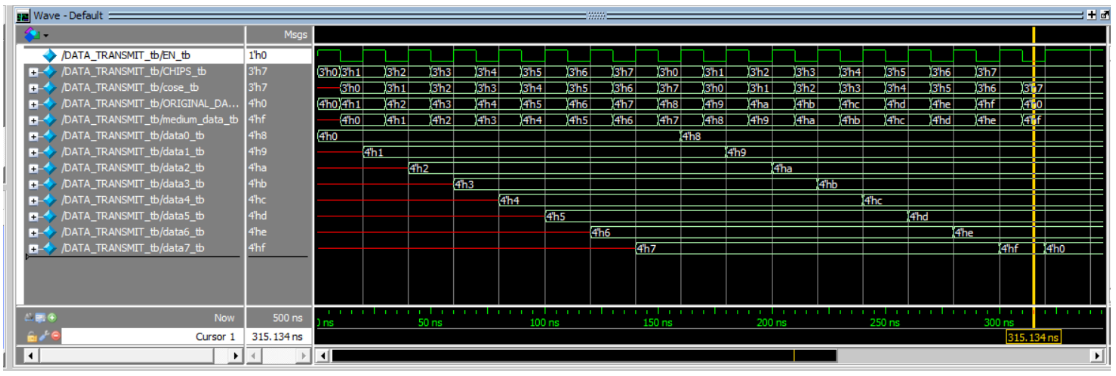
```
module Four_Seven_Decode_LED(medium_data,a,b,c,d,e,f,g);
    input [3:0]medium_data;
    output a,b,c,d,e,f,g;
    reg [6:0]final_data;

    always@(medium_data)begin
        case(medium_data)
            4'b0000: final_data<=7'b0000001;
            4'b0001: final_data<=7'b1001111;
            4'b0010: final_data<=7'b0010010;
            4'b0011: final_data<=7'b0000110;
            4'b0100: final_data<=7'b1001100;
            4'b0101: final_data<=7'b0100100;
            4'b0110: final_data<=7'b0100000;
            4'b0111: final_data<=7'b0001111;
            4'b1000: final_data<=7'b0000000;
            4'b1001: final_data<=7'b0000100;
            4'b1010: final_data<=7'b0001000;
            4'b1011: final_data<=7'b1100000;
            4'b1100: final_data<=7'b0110001;
            4'b1101: final_data<=7'b1000010;
            4'b1110: final_data<=7'b0110000;
            4'b1111: final_data<=7'b0111000;
        endcase
    end
    assign a=final_data[6];
    assign b=final_data[5];
    assign c=final_data[4];
    assign d=final_data[3];
    assign e=final_data[2];
    assign f=final_data[1];
    assign g=final_data[0];

endmodule
```

四七译码器

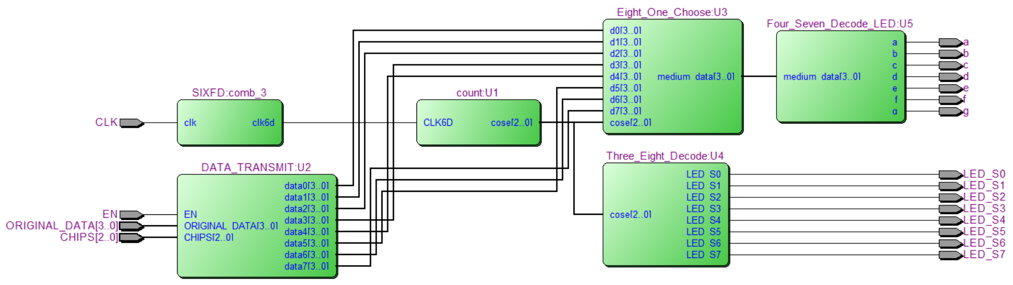
仿真

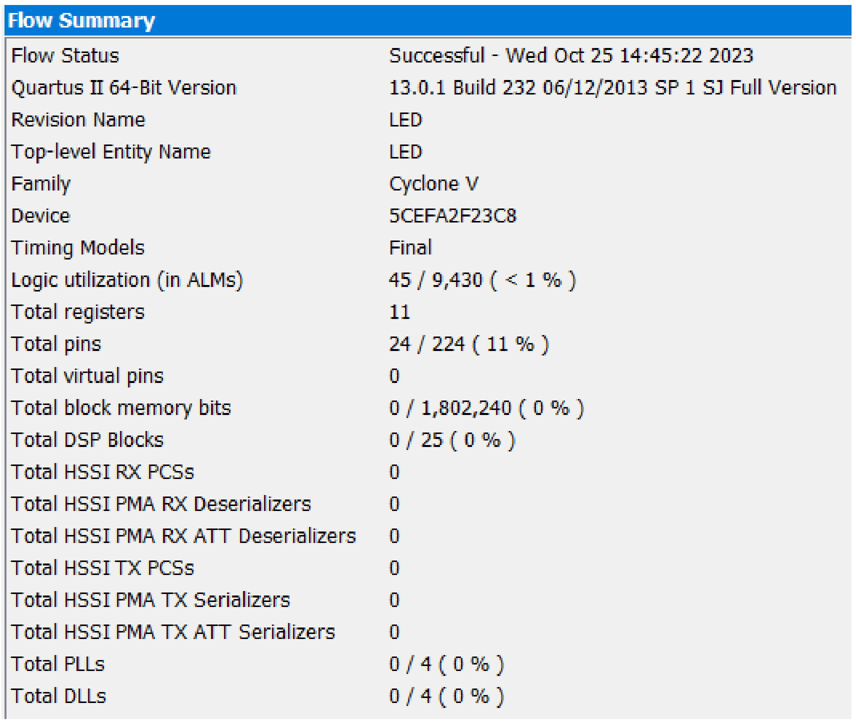


```
`timescale 10ns/1ns
module DATA_TRANSMIT_tb();
    reg EN_tb;
    reg [2:0] CHIPS_tb,cose_tb;
    reg [3:0] ORIGINAL_DATA_tb;
    wire [3:0] medium_data_tb,data0_tb,data1_tb,data2_tb,data3_tb,data4_tb,data5_tb,data6_tb,data7_tb;

    DATA_TRANSMIT U0(EN_tb,ORIGINAL_DATA_tb,CHIPS_tb,data0_tb,data1_tb,data2_tb,data3_tb,data4_tb,data5_tb,data6_tb,data7_tb);

    initial begin
    #0 EN_tb=1'b1;#0 CHIPS_tb=3'b000;#0 ORIGINAL_DATA_tb=4'b0000;
    #1 EN_tb=1'b0;#0 cose_tb=3'b000;#0 CHIPS_tb=3'b001;#0 ORIGINAL_DATA_tb=4'b0001;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b001;#0 CHIPS_tb=3'b010;#0 ORIGINAL_DATA_tb=4'b0010;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b010;#0 CHIPS_tb=3'b011;#0 ORIGINAL_DATA_tb=4'b0011;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b011;#0 CHIPS_tb=3'b100;#0 ORIGINAL_DATA_tb=4'b0100;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b100;#0 CHIPS_tb=3'b101;#0 ORIGINAL_DATA_tb=4'b0101;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b101;#0 CHIPS_tb=3'b110;#0 ORIGINAL_DATA_tb=4'b0110;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b110;#0 CHIPS_tb=3'b111;#0 ORIGINAL_DATA_tb=4'b0111;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b111;#0 CHIPS_tb=3'b000;#0 ORIGINAL_DATA_tb=4'b1000;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b000;#0 CHIPS_tb=3'b001;#0 ORIGINAL_DATA_tb=4'b1001;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b001;#0 CHIPS_tb=3'b010;#0 ORIGINAL_DATA_tb=4'b1010;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b010;#0 CHIPS_tb=3'b011;#0 ORIGINAL_DATA_tb=4'b1011;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b011;#0 CHIPS_tb=3'b100;#0 ORIGINAL_DATA_tb=4'b1100;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b100;#0 CHIPS_tb=3'b101;#0 ORIGINAL_DATA_tb=4'b1101;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b101;#0 CHIPS_tb=3'b110;#0 ORIGINAL_DATA_tb=4'b1110;#1 EN_tb=1'b1;
    #1 EN_tb=1'b0;#0 cose_tb=3'b110;#0 CHIPS_tb=3'b111;#0 ORIGINAL_DATA_tb=4'b1111;#1 EN_tb=1'b1;

    #1 EN_tb=1'b0;#0 cose_tb=3'b111;#0 CHIPS_tb=3'b111;#0 cose_tb=3'b111;#0 ORIGINAL_DATA_tb=4'b0000;#1 EN_tb=1'b1;
    end

    Eight_One_Choose U1(data0_tb,data1_tb,data2_tb,data3_tb,data4_tb,data5_tb,data6_tb,data7_tb,cose_tb,medium_data_tb);


endmodule
```


Wave - Default
Msgs
/DATA_TRANSMIT_tb/EN_tb
/DATA_TRANSMIT_tb/CHIPS_tb
/DATA_TRANSMIT_tb/cose_tb
/DATA_TRANSMIT_tb/ORIGINAL_DA...
/DATA_TRANSMIT_tb/medium_data_tb
/DATA_TRANSMIT_tb/data0_tb
/DATA_TRANSMIT_tb/data1_tb
/DATA_TRANSMIT_tb/data2_tb
/DATA_TRANSMIT_tb/data3_tb
/DATA_TRANSMIT_tb/data4_tb
/DATA_TRANSMIT_tb/data5_tb
/DATA_TRANSMIT_tb/data6_tb
/DATA_TRANSMIT_tb/data7_tb
Now 500 ns
Cursor 1 315.134 ns
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
315.134 ns

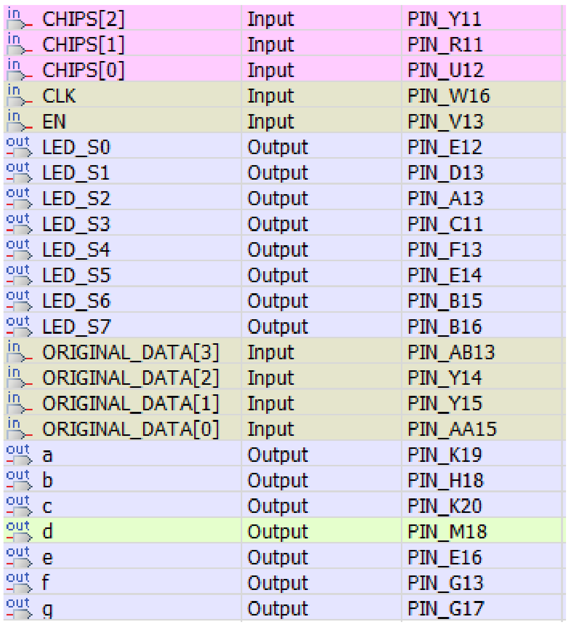
中间结果





Flow Summary	
Flow Status	Successful - Wed Oct 25 14:45:22 2023
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	LED
Top-level Entity Name	LED
Family	Cyclone V
Device	5CEFA2F23C8
Timing Models	Final
Logic utilization (in ALMs)	45 / 9,430 (< 1 %)
Total registers	11
Total pins	24 / 224 (11 %)
Total virtual pins	0
Total block memory bits	0 / 1,802,240 (0 %)
Total DSP Blocks	0 / 25 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI PMA RX ATT Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total HSSI PMA TX ATT Serializers	0
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 4 (0 %)

FPGA 验证结果



in CHIPS[2]	Input	PIN_Y11
in CHIPS[1]	Input	PIN_R11
in CHIPS[0]	Input	PIN_U12
in CLK	Input	PIN_W16
in EN	Input	PIN_V13
out LED_S0	Output	PIN_E12
out LED_S1	Output	PIN_D13
out LED_S2	Output	PIN_A13
out LED_S3	Output	PIN_C11
out LED_S4	Output	PIN_F13
out LED_S5	Output	PIN_E14
out LED_S6	Output	PIN_B15
out LED_S7	Output	PIN_B16
in ORIGINAL_DATA[3]	Input	PIN_AB13
in ORIGINAL_DATA[2]	Input	PIN_Y14
in ORIGINAL_DATA[1]	Input	PIN_Y15
in ORIGINAL_DATA[0]	Input	PIN_AA15
out a	Output	PIN_K19
out b	Output	PIN_H18
out c	Output	PIN_K20
out d	Output	PIN_M18
out e	Output	PIN_E16
out f	Output	PIN_G13
out g	Output	PIN_G17

改变 CHIPS 信号可以选择要设定的位,改变 ORIGINAL_DATA 信号可以改变该位置的数字, EN 是使能信号, EN=1 时使能, EN=0 时保持。

总结

注意顶层实体数据传输的过程中数据位数不要写错