Code 全体代码，全体起立

数据选择器

```
//二选一数据选择器 assign型
module MUX2_1_assign(a,b,s,y);
  input a,b,s;
  output y;

  assign y=(s==0?)a:b;
endmodule

//二选一数据选择器 always-if型
module MUX2_1_if(a,b,s,y);
  input a,b,s;
  output y;
  reg y;

  always@(a,b,s)
  begin
    if(!s)
    y=a;
    else
    y=b;
  end

endmodule

//四选一数据选择器 assign型
module MUX4_1_assign(d0,d1,d2,d3,s1,s2,y);
  input d0,d1,d2,d3,s1,s2;
  output y;
  reg y;

  wire[1:0] SEL;
  wire d0T,d1T,d2T,d3T;

  assign SEL={s1,s0};
  assign d0T=(SEL==2'b00);
  assign d1T=(SEL==2'b01);
  assign d2T=(SEL==2'b10);
  assign d3T=(SEL==2'b11);

  assign
y=(d0&d0T)|(d1&d1T)|(d2&d2T)|(d3&d3T);

endmodule

//四选一数据选择器 case型
module MUX4_1_case(d0,d1,d2,d3,s1,s2,y);
  input d0,d1,d2,d3,s1,s2;
  output y;
  reg y;

  always@(d0,d1,d2,d3,s1,s0)
  begin
    case({s1,s0})
    2'b00: y<=d0;
    2'b01: y<=d1;
    2'b10: y<=d2;
    2'b11: y<=d3;
    default: y<=d0;
    endcase
  end

endmodule

//四选一数据选择器 if型
module MUX4_1_if(d0,d1,d2,d3,s1,s0,y);
  input d0,d1,d2,d3,s1,s0;
  output y;
  reg y;

  always@(d0,d1,d2,d3,s1,s0)
  begin
    if(!s0)
  begin
    if(!s1) y<=d0;
    else y<=d1;
  end
  else begin
    if(!s1) y<=d2;
    else y<=d3;
  end
  end

endmodule

//八选一数据选择器
module
MUX8_1(d0,d1,d2,d3,d4,d5,d6,d7,SEL,Out);
  input d0,...d7;
  input[2:0] SEL;
  output Out;
  reg Out;
  always@(d0,...d7,SEL)
  begin
    case(SEL)
    3'b000: Out=d0;
    ...
    3'b111: Out=d7;
    default: Out=1'bx;
    endcase
  end

endmodule
```

加法器

```
//半加器
module half_adder(a,b,S0,C0)
  input a,b;
  output S0,C0;
  assign S0=a^b;
  assign C0=a&b;
endmodule

//全加器(非例化)
module full_adder(a,b,Cin,S0,C0);
  input a,b,Cin;
  output S0,C0;
  reg s1,s2,s3;

  always@(a,b,Cin)
  begin
    S0=(a^b)^Cin;
    s1=a&b;
    s2=a&Cin;
    s3=b&Cin;
    C0=(s1|s2)|s3;
  end

endmodule

//全加器(半加器例化)
module full_adder_half(a,b,Cin,S0,C0);
  input a,b,Cin;
  output S0,C0;
  reg S0,C0;

  wire first_S0, first_C0, second_C0;

  half_adder first_adder(a,b,first_S0,first_C0);
  half_adder second_adder(first_S0,Cin,S0,second_C0);

  assign C0=first_C0|second_C0;

endmodule

//八位加法器(非例化)
module Adder_8b(a,b,Cin,S0,C0);
  input[7:0] a,b;
  input Cin;
  output[7:0] S0;
  ourput C0;

  wire[8:0] DATA;

  assign DATA=a+b+Cin;
  assign C0=DATA[8];
  assign S0=DATA[7:0];

endmodule

//八位加法器(非例化的全加器例化)
module adder_8bit_full(a,b,Cin,S0,C0);
  input[7:0] a,b;
  input Cin;
  output[7:0] S0;
  output C0;

  wire c1,c2...c7;

  full_adder u0(a[0],b[0],Cin,c1,S0[0]);
  full_adder u1(a[1],b[1],c1,c2,S0[1]);
  ...
  full_adder u7(a[8],b[8],c7,C0,S0[7]);

endmodule

//四位加法器(非例化)
module Adder_4bit(a,b,Cin,S0,C0);
  input[3:0] a,b;
  input Cin;
  output[3:0] S0;
  output C0;

  wire[4:0] DATA;

  assign DATA=a+b+Cin;
  assign C0=DATA[4];
  assign S0=DATA[3:0];

endmodule

//四位加法器(非例化的全加器例化)
module adder_4bit_full(a,b,Cin,S0,C0);
  input[3:0] a,b;
  input Cin;
  output[3:0] S0;
  output C0;

  wire c1,c2,c3;

  full_adder u0(a[0],b[0],Cin,c1,S0[0]);
  full_adder u1(a[1],b[1],c1,c2,S0[1]);
  ...
  full_adder u7(a[3],b[3],c3,C0,S0[3]);

endmodule
```

计数器

```
//十进制计数器(无使能同步)
module CNT10(CLK,RST,Q,C0)
  input CLK,RST;
  output C0;
  output[3:0] Q;
  reg[3:0] Q;

  always@(posedge CLK)
  begin
    if(!RST) Q<=0;
    else if(Q>=4'd9) Q<=0;
    else Q<=Q+1;
  end

  assign C0=(Q==4'd9)?1'b1:1'b0;
endmodule

//十进制计数器(使能同步)
module CNT10(CLK,RST,EN,Q,C0)
  input CLK,RST,EN;
  output C0;
  output[3:0] Q;
  reg[3:0] Q;

  always@(posedge CLK)
  begin
    if(!RST) Q<=0;
    else if(EN)
    begin
      if(Q>=4'd9) Q<=0;
      else Q<=Q+1;
    end
  end

  assign C0=(Q==4'd9)?1'b1:1'b0;
endmodule

//十进制计数器(使能带重异步)
module CNT10(CLK,RST,DATA,LOAD,EN,Q,C0)
  input CLK,RST,EN,LOAD;
  input[3:0] DATA;
  output C0;
  output[3:0] Q;
  reg[3:0] Q;

  always@(posedge CLK , negedge RST)
  begin
    if(!RST) Q<=0;
    else if(EN)
    begin
      if(LOAD) Q<=DATA;
      else if(Q>=4'd9) Q<=0;
      else Q<=Q+1;
    end
  end

  assign C0=(Q==4'd9)?1'b1:1'b0;
endmodule

//60进制计数器(使能异步非例化)
module CNT60(CLK,RST,EN,Q,C0);
  input CLK,RST,EN;
  output C0;
  output[3:0] Q;
  reg[3:0] Q;

  always@(posedge CLK , negedge RST)
  begin
    if(RST) Q<=6'd0;
    else if(EN)
    begin
      if(Q>=6'd59) Q<=6'd0;
      else Q<=Q+1;
    end
  end

endmodule

//6进制计数器
module CNT6(CLK,RST,Q,C0)
  input CLK,RST;
  output C0;
  output[2:0] Q;
  reg[2:0] Q;

  always@(posedge CLK)
  begin
    if(!RST) Q<=0;
    else if(Q==3'd6) Q<=0;
    else Q<=Q+1;
  end

  assign C0=(Q==3'd6)?1'b1:1'b0;
endmodule

//60进制计数器(例化)
module CNT60(CLK,RST,EN,Q1,Q0,C0);
  input CLK,RST,EN;
  output[3:0]Q1,Q0;
  output C0;

  wire C0_0,Codd1,C0_1;
  CNT10 U1(CLK,RST,EN,Q0,C0_0);
  not U1_2(Codd1,C0_0);
  CNT6 U2(Codd1,RST,EN,Q1,C0_1);
  and(C0,C0_0,C0_1);
endmodule

//100进制计数器(例化)
module CNT100(CLK,RST,EN,Q1,Q0,C0);
  input CLK,RST,EN;
  output[3:0]Q1,Q0;
  output C0;

  wire C0_0,Codd1,C0_1;
  CNT10 U1(CLK,RST,EN,Q0,C0_0);
  not U1_2(Codd1,C0_0);
  CNT10 U2(Codd1,RST,EN,Q1,C0_1);
  and(C0,C0_0,C0_1);
endmodule

//1000进制计数器(例化)
module CNT1000(CLK,RST,EN,Q2,Q1,Q0,C0)
  input CLK,RST,EN;
  output[3:0]Q2,Q1,Q0;
  output C0;

  wire C0_0,Codd,C0_1,Codd1,C0_2;
  CNT10 U1(CLK,RST,EN,Q0,C0_0);
  not U1_2(Codd,C0_0);
  CNT10 U2(Codd,RST,EN,Q1,C0_1)
  not U2_3(Codd1,C0_1);
  CNT10 U3(Codd1,RST,EN,Q2,C0_2);
  and(C0,C0_0,C0_1,C0_2);
endmodule
```

乘法器

```
//四位乘法器(for增值)
module MULT_4(R,A,B);
  parameter S=4;
  input[S:1] A,B;
  output[2*S:1] R;

  integer i;
  reg[2*S:1]  R;

  always@(A,B)
  begin
    R=0;
    for(i=1;i<=S;i=i+1)
      if(B[i]) R=R+(A<<(i-1));
      else R=R;
  end
endmodule

//两位乘法器(for增值)
module MULT_2(R,A,B);
  parameter S=2;
  input[S:1] A,B;
  output[2*S:1] R;

  integer i=1;
  reg[2*S:1]  R;

  always@(A,B)
  begin
    R=0;
    for(i=1;i<=S;i=i+1)
      if(B[i]) R=R+(A<<(i-1));
      else R=R;
  end

endmodule
```

分频器

```
module div_1Hz(clk,dout);
  input clk;
  output dout;
  reg dout;
  integer i;
  always @(posedge clk)
    begin
    i=i+1;
    if(i==24999999) //5MHz——1Hz
      begin
      dout=~dout;
      i=0;
      end
    end
endmodule
```

序列检测器

```
//序列检测器
module SCHK(CLK,IN,RST,OUT);
  parameter s0=4'd1,s1=4'd...s8=4'd8;
  reg[8:0]NOW,NEX;

  always@(posedge CLK , negedge RST)
  begin
    if(RST) NOW<=s0;
    else NOW<=NEX;

    always@(NOW,NEX)
    begin
      case(NOW)
      s0: if(IN=1'b1) NEX<=s1; else NEX<=s0;
      s1: if(IN=1'b1) NEX<=s2; else NEX<=s0;
      ...
      s8: if(IN=1'b0) NEX<=s3; else NEX<=s2;
      default:NEX<=s0;
      endcase
    end
  end
  assign OUT=(NOW==s8);

endmodule
```

奇偶校验

```
//奇偶校验模块
module odd(DATA,sel,S);
  input[8:0] DATA;
  input sel;
  output S;

  wire odd;
  assign odd=^DATA;
  assign S=sel?odd:~odd;

endmodule
```

移位寄存器

```
//移位寄存器
module REG(CLK,LOAD,DATA,Q);
  input CLK,LOAD
  input[7:0] DATA;
  output Q;

  reg[7:0]  r;
  always@(posedge CLK)
  begin
    if(LOAD) r<=DATA;
    else r[6:0]<=r[7:1];
    assign Q=r[0];
  end

endmodule
```