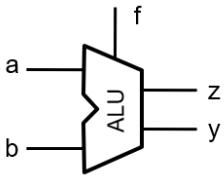
Lab01-运算器及其应用 实验报告

实验过程

32bit ALU

1. Datapath & function



ALU 模块功能表

f	y	z
000	a + b	*
001	a - b	*
010	a & b	*
011	a | b	*
100	a ^ b	*
其他	0	1

2. Interface

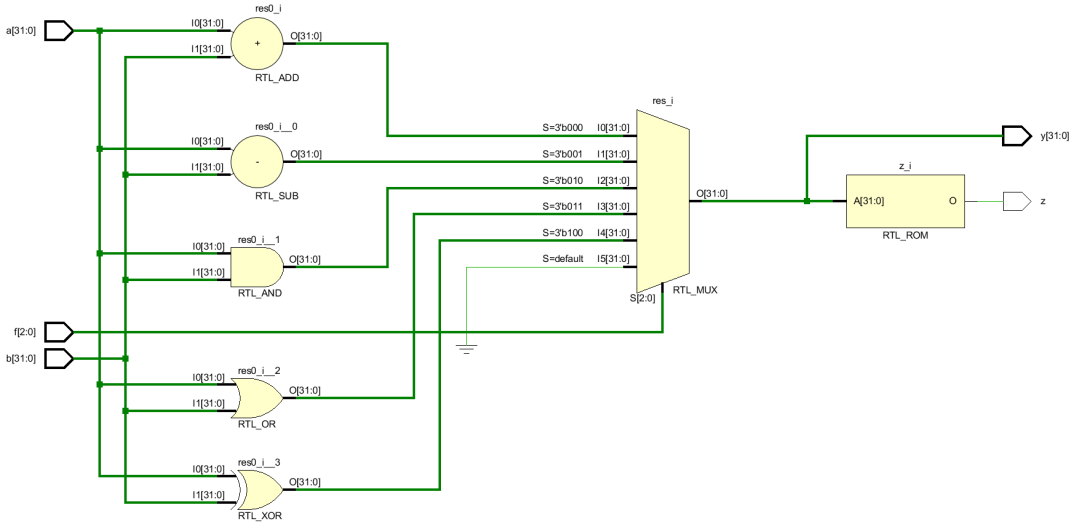
```
module ALU #(
parameter WIDTH = 32     //数据宽度
)(
input [WIDTH-1:0] a, b, //两操作数
input [2:0] f,          //操作功能
output [WIDTH-1:0] y,   //运算结果
output z                //零标志
);
```

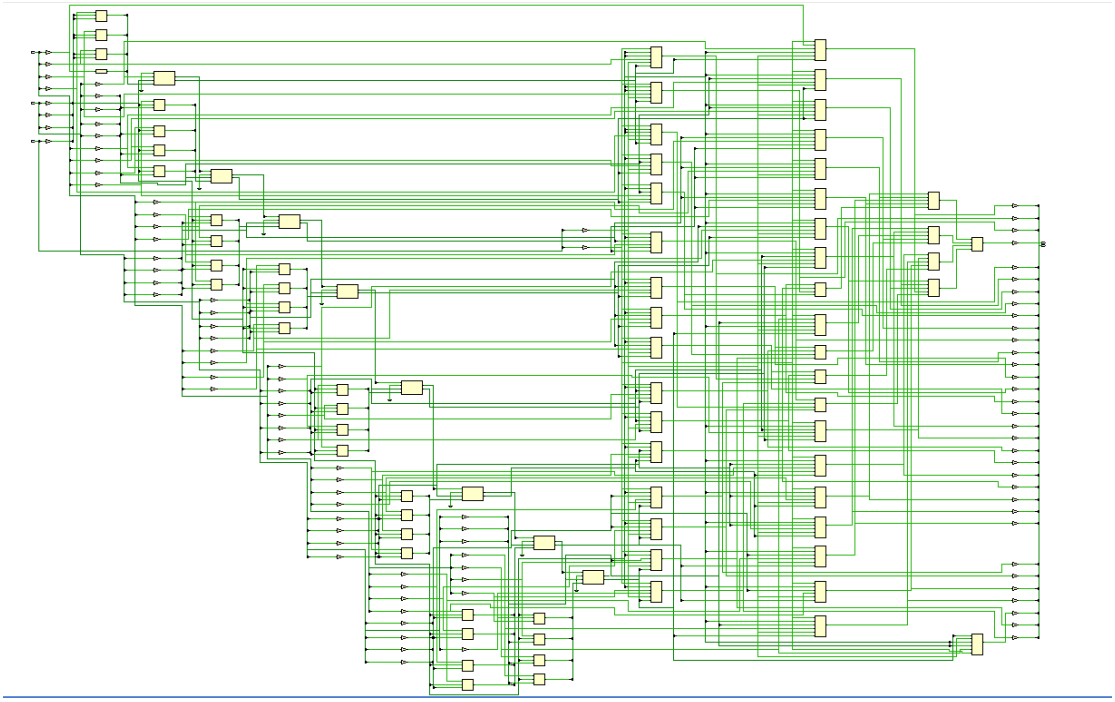
3. 核心

```
assign z = y == 32'b0 ? 1:0;

always @(*) begin
    case (f)
    3'b000: res = a + b;
    3'b001: res = a - b;
    3'b010: res = a & b;
    3'b011: res = a | b;
    3'b100: res = a ^ b;
    default: res = 0;
    endcase
end
```

4. 电路





5. 性能

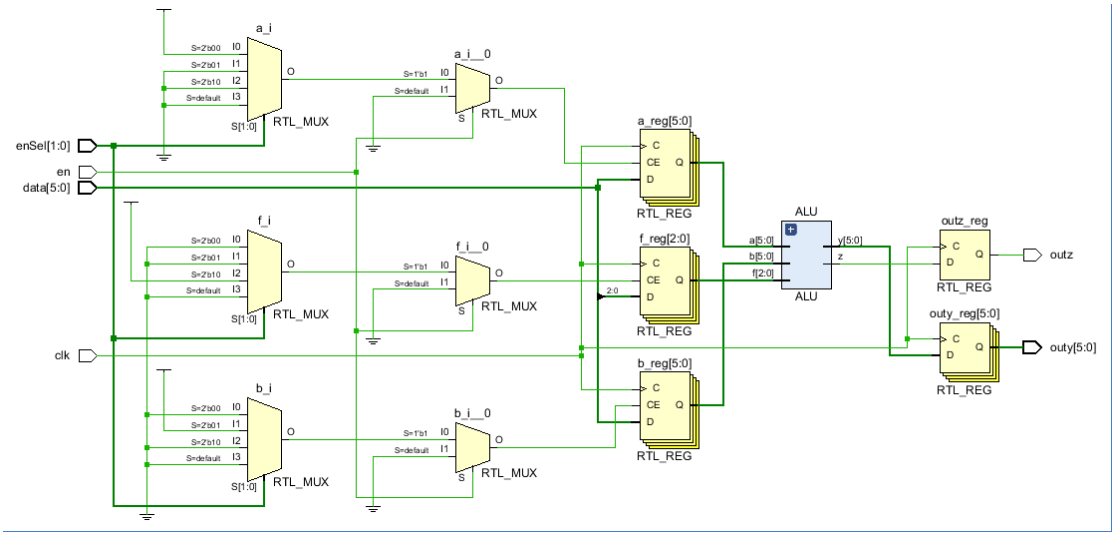
Name 1	Slice LUTs (63400)	Bonded IOB (210)
N ALU	74	100

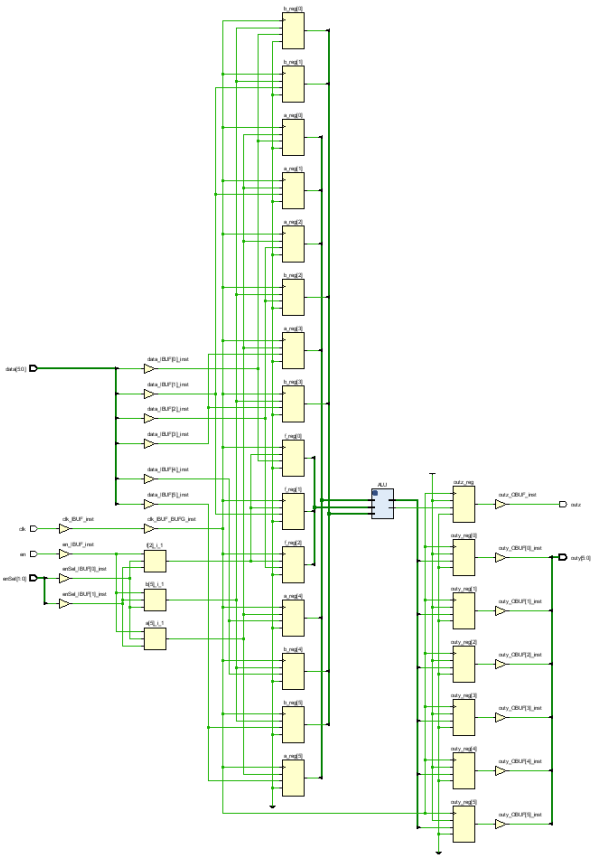
6bit ALU

1. Code

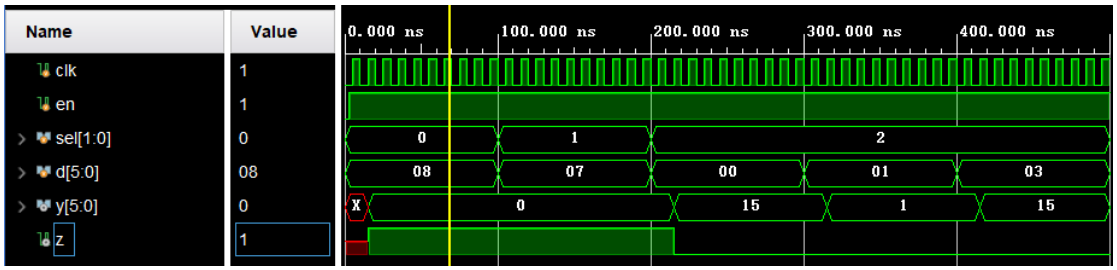
```verilog
module ALU6b(
input [5:0] data,
input en,
input [1:0] enSel,
input clk,
output reg outz,
output reg [5:0] outy
);
reg [5:0] a, b;
reg [2:0] f;
wire [5:0] y;
wire z;
ALU #(6)ALU(.a(a),.b(b),.f(f),.y(y),.z(z));
    always @(posedge clk) begin
        if(en) begin
            case (enSel)
            2'b00: a <= data;
            2'b01: b <= data;
            2'b10: f <= {data[2],data[1],data[0]};
            endcase
        end
    outz <= z;
    outy <= y;
    end
endmodule
```

2. RTL & 综合仿真





3. 仿真

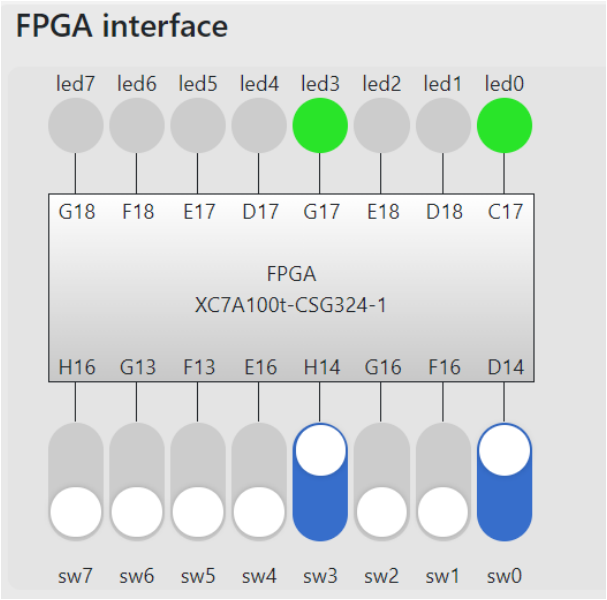


4. 性能

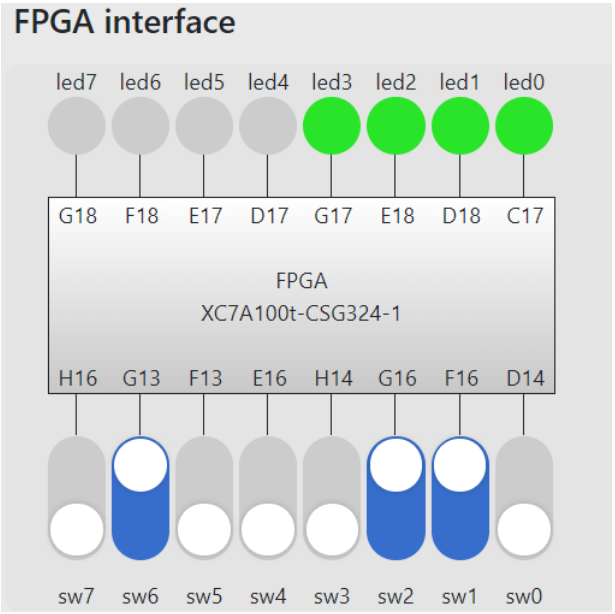
Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
N ALU6b	16	22	17	1
ALU (ALU)	13	0	0	0

5. 下载测试

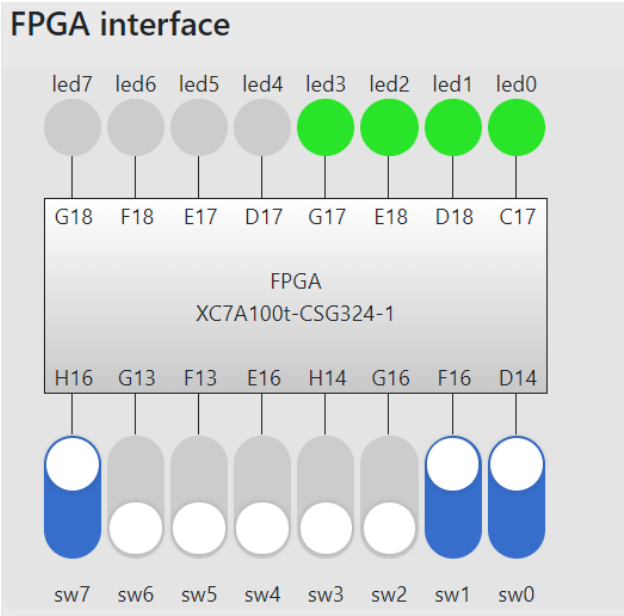
1. 输入 a



2. 输入 b（此时 f 初始为 0，加运算）



3. 输入 f（与运算）



FLS

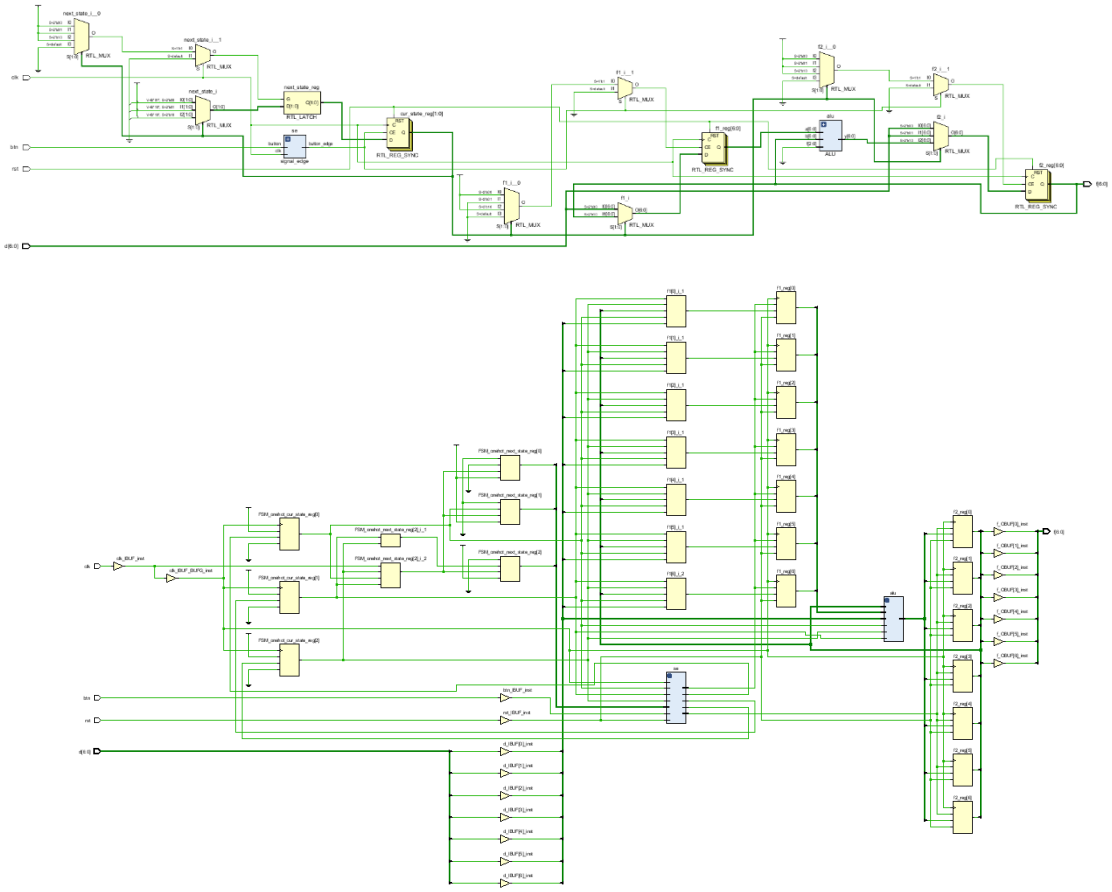
1. Code

```
always @(*) begin
    if(clk)begin
        case(cur_state)
        GET_D0: next_state = GET_D1;
        GET_D1: next_state = CAL_F;
        CAL_F: next_state = CAL_F;
        endcase
    end
end
```

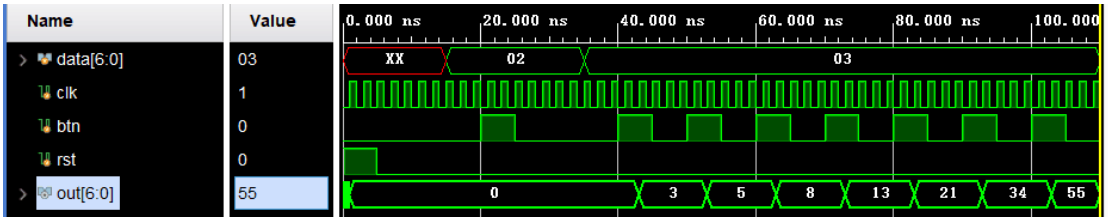
```
assign f = f2;
```

```verilog
always @(posedge clk) begin
    if(rst) begin
        cur_state <= GET_D0;
        f1<=7'b0;
        f2<=7'b0;
    end
    else begin
        if(en)begin
            case (cur_state)
            GET_D0: begin
                f1 <= d; f2 <= d;
            end
            GET_D1: f2 <= d;
            CAL_F: begin
                f2 <= sum;
                f1 <= f2;
            end
            endcase
            cur_state <= next_state;
        end
    end
end
```

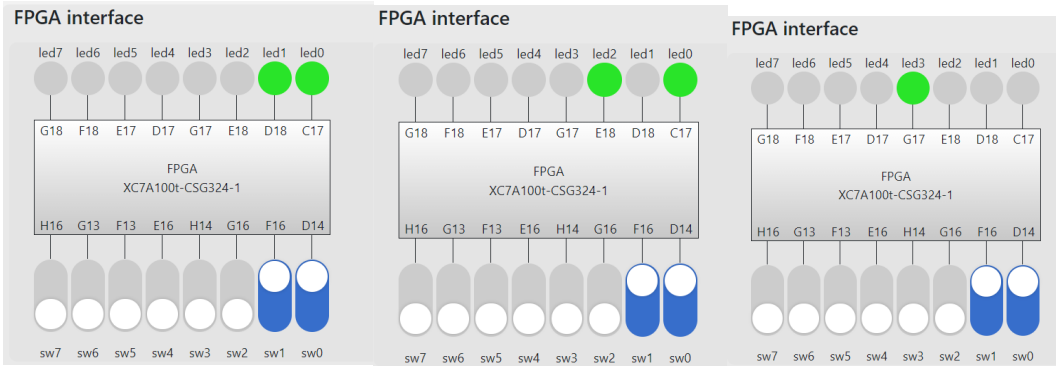
2. 电路



3. 仿真



4. 下载测试



总结

本次实验本身难度较低，给同学们提供了一个回忆 verilog 的机会。再实验过程中回忆、巩固了 verilog 相关的基本知识。

Appendix

代码

1. ALU

```
module ALU #(
parameter WIDTH = 32     //数据宽度
)(
input [WIDTH-1:0] a, b, //两操作数
input [2:0] f,          //操作功能
output [WIDTH-1:0] y,   //运算结果
output z                //零标志
);

reg [WIDTH-1:0] res;
```

```
assign y = res;
assign z = y == 32'b0 ? 1:0;

always @(*) begin
    case (f)
    3'b000: res = a + b;
    3'b001: res = a - b;
    3'b010: res = a & b;
    3'b011: res = a | b;
    3'b100: res = a ^ b;
    default: res = 0;
    endcase
end
endmodule
```

2. 6bit ALU

```
module ALU6b(
input [5:0] data,
input en,
input [1:0] enSel,
input clk,
output reg outz,
output reg [5:0] outy
);
reg [5:0] a, b;
reg [2:0] f;
wire [5:0] y;
wire z;
ALU #(6)ALU(.a(a),.b(b),.f(f),.y(y),.z(z));
    always @(posedge clk) begin
        if(en) begin
            case (enSel)
            2'b00: a <= data;
            2'b01: b <= data;
            2'b10: f <= {data[2],data[1],data[0]};
            endcase
        end
    outz <= z;
    outy <= y;
```

```
    end
endmodule
```

3. FLS

```
module  FLS(
  input  clk, rst, btn,
  input  [6:0]  d,
  output [6:0]  f
);

parameter GET_D0 = 2'b00; // To get the 1st number
parameter GET_D1 = 2'b01; // To get the 2nd number
parameter CAL_F = 2'b10;  // To calculate the next
number

wire en;

signal_edge se(clk,btn,en);

reg[1:0] cur_state, next_state;
reg[6:0] f1, f2;
wire[6:0] sum;

ALU #(7)alu(.a(f1),.b(f2),.f(3'b000), .y(sum));


always @(*) begin
    if(clk)begin
        case(cur_state)
        GET_D0: next_state = GET_D1;
        GET_D1: next_state = CAL_F;
        CAL_F: next_state = CAL_F;
        endcase
    end
end

always @(posedge clk) begin
    if(rst) begin
        cur_state <= GET_D0;
        f1<=7'b0;
```

```
        f2<=7'b0;
    end
    else begin
        if(en)begin
            case (cur_state)
            GET_D0: begin
                f1 <= d; f2 <= d;
            end
            GET_D1: f2 <= d;
            CAL_F: begin
                f2 <= sum;
                f1 <= f2;
            end
            endcase
            cur_state <= next_state;
        end
    end
end

assign f = f2;

endmodule

// 取时钟上升沿
module signal_edge(
    input clk,
    input button,
    output button_edge
    );
    reg button_r1,button_r2;
    always@(posedge clk)
        button_r1 <= button;
    always@(posedge clk)
        button_r2 <= button_r1;
    assign button_edge = button_r1 & (~button_r2);
endmodule
```

4. 6bit ALU 仿真

```
module ALU6_test();
```

```
    reg clk,en;
    reg [1:0] sel;
    reg [5:0] d;
    wire [5:0] y;
    wire z;
    ALU6b ALU6b(.clk(clk),.en(en),.enSel(sel),.data
(d),.outy(y),.outz(z));

    initial clk=0;
    always #5 clk=~clk;

    initial
    begin
        en=1'b0;
        #3 en=1'b1;
    end
    // always #1 en=~en;

    initial
    begin
            sel=2'b00;d=6'b01000;
        #100 sel=2'b01;d=6'b000111;
        #100 sel=2'b10;d=6'b000000;
        #100 d=6'b01;
        #100 d=6'b11;
        #100 $finish;
    end
endmodule
```

5. FLS 仿真

```
module fls_testbench();
    reg[6:0] data;
    reg clk,btn,rst;
    wire[6:0] out;
    fls fls(clk,rst,btn,data,out);
    initial clk = 0;
    always #1 clk=~clk;

    initial
```

```
    begin
        rst = 1; btn = 0;
    #5 rst = 0;
    #10 data = 7'd2;
    #5  btn = 1; #5  btn =0;
    #10 data = 7'd3;
    #5  btn = 1; #5  btn =0;
    #5  btn = 1; #5  btn =0;
    #5  btn = 1; #5  btn =0;
    #5  btn = 1; #5  btn =0;
    #5  btn = 1; #5  btn =0;
    #5  btn = 1; #5  btn =0;
    #5  btn = 1; #5  btn =0;
    #5 $finish;
    end
endmodule
```