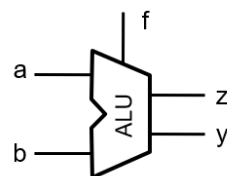


# Lab01-运算器及其应用 实验报告

## 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	$a \wedge b$	*
其他	0	1

#### 2. Interface

```
1 module ALU #(
2   parameter WIDTH = 32    //数据宽度
3 )(
4   input [WIDTH-1:0] a, b,  //两操作数
5   input [2:0] f,           //操作功能
6   output [WIDTH-1:0] y,    //运算结果
7   output z                 //零标志
8 );
```

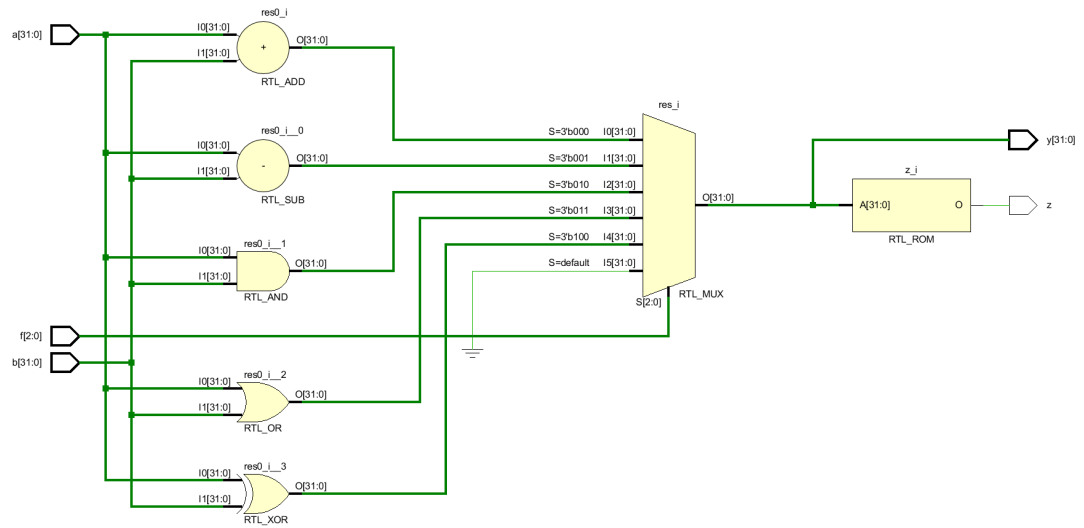
#### 3. 核心

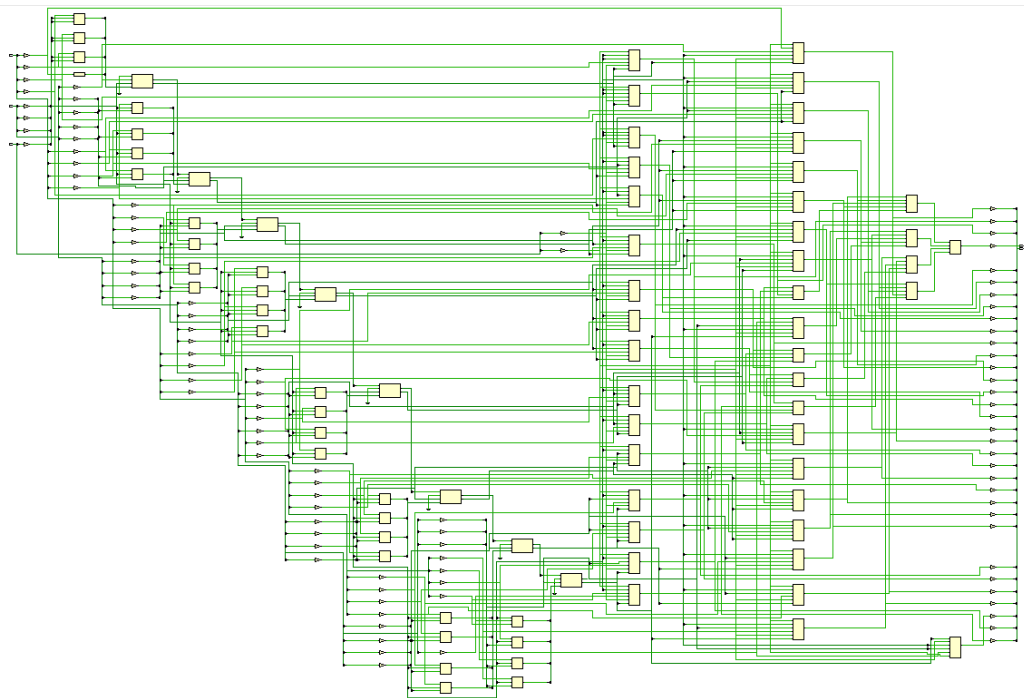
```

1  assign z = y == 32'b0 ? 1:0;
2
3  always @(*) begin
4      case (f)
5          3'b000: res = a + b;
6          3'b001: res = a - b;
7          3'b010: res = a & b;
8          3'b011: res = a | b;
9          3'b100: res = a ^ b;
10         default: res = 0;
11     endcase
12 end

```

#### 4. 电路





5. 性能

Name <sup>1</sup>	Slice LUTs (63400)	Bonded IOB (210)
N ALU	74	100

## 6bit ALU

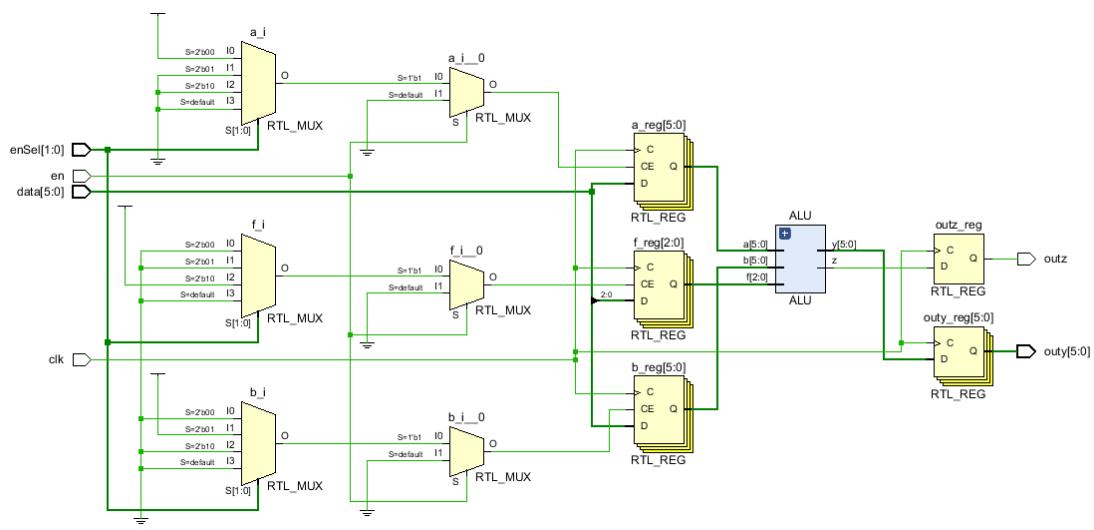
1. Code

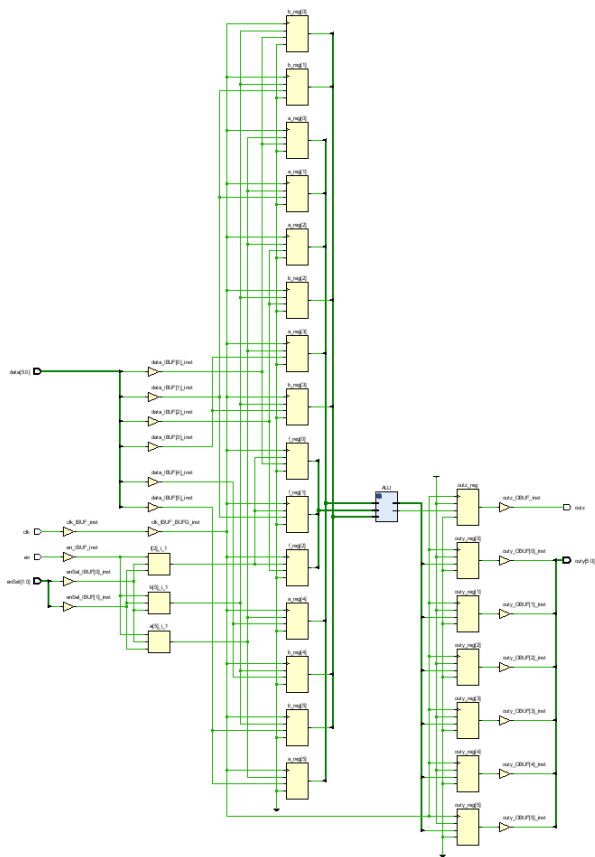
```

1  module ALU6b(
2  input [5:0] data,
3  input en,
4  input [1:0] enSel,
5  input clk,
6  output reg outz,
7  output reg [5:0] outy
8  );
9  reg [5:0] a, b;
10 reg [2:0] f;
11 wire [5:0] y;
12 wire z;
13 ALU #(6)ALU(.a(a),.b(b),.f(f),.y(y),.z(z));
14     always @(posedge clk) begin
15         if(en) begin
16             case (enSel)
17                 2'b00: a ≤ data;
18                 2'b01: b ≤ data;
19                 2'b10: f ≤ {data[2],data[1],data[0]};
20             endcase
21         end
22         outz ≤ z;
23         outy ≤ y;
24     end
25 endmodule

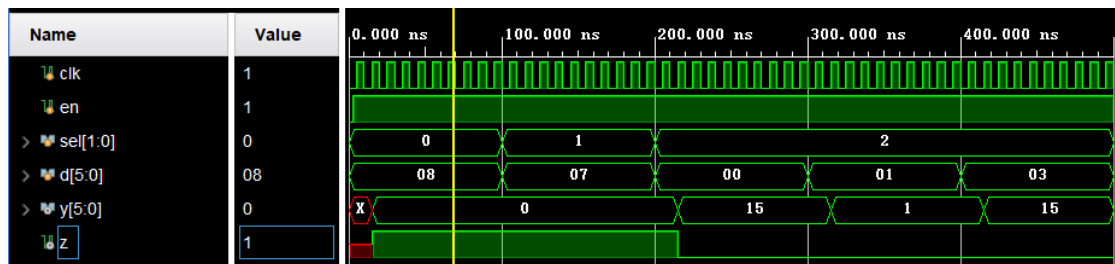
```

## 2. RTL & 综合仿真





### 3. 仿真



### 4. 性能

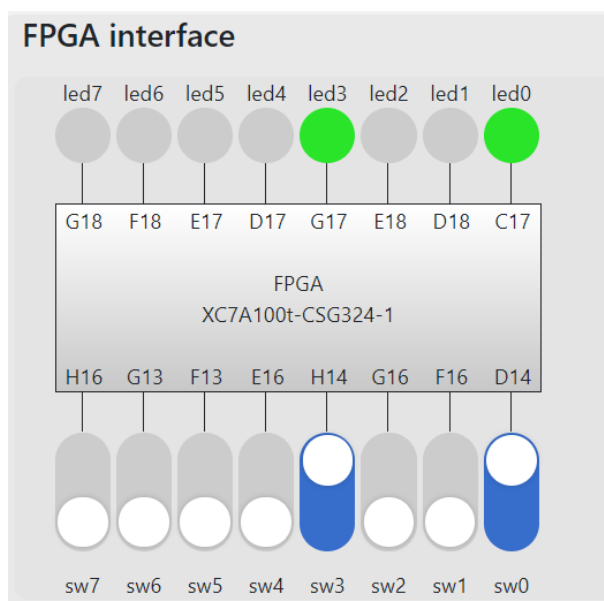
Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
ALU6b	16	22	17	1
ALU (ALU)	13	0	0	0

Design Timing Summary			
Setup	Hold	Pulse Width	
Worst Negative Slack (WNS): 6.119 ns	Worst Hold Slack (WHS): 0.213 ns	Worst Pulse Width Slack (WPWS): 4.500 ns	
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns	
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	
Total Number of Endpoints: 7	Total Number of Endpoints: 7	Total Number of Endpoints: 23	

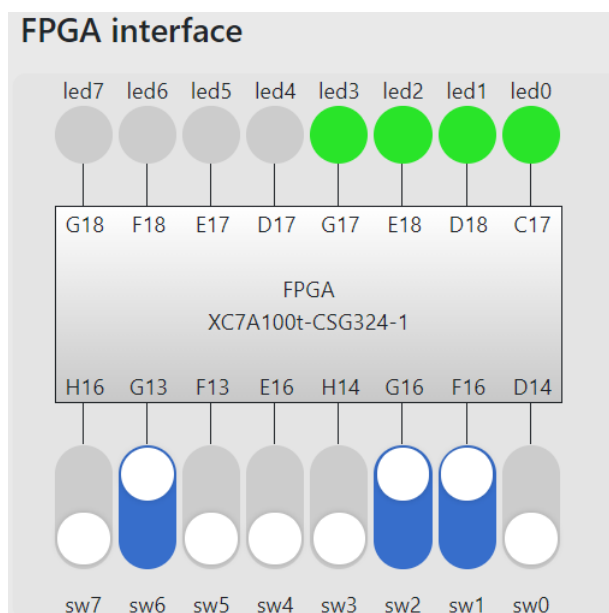
All user specified timing constraints are met.

### 5. 下载测试

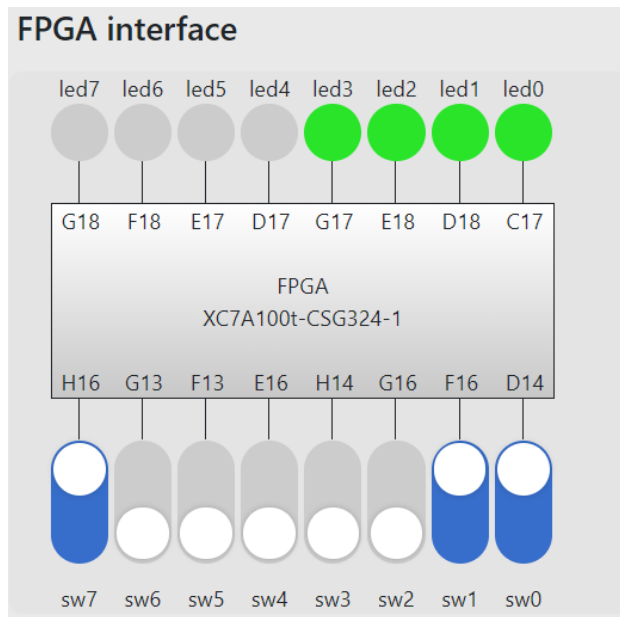
#### 1. 输入 a



2. 输入 b (此时 f 初始为 0, 加运算)



3. 输入 f (与运算)



## FLS

### 1. Code

```
1 always @(*) begin
2     if(clk)begin
3         case(cur_state)
4             GET_D0: next_state = GET_D1;
5             GET_D1: next_state = CAL_F;
6             CAL_F: next_state = CAL_F;
7         endcase
8     end
9 end
```

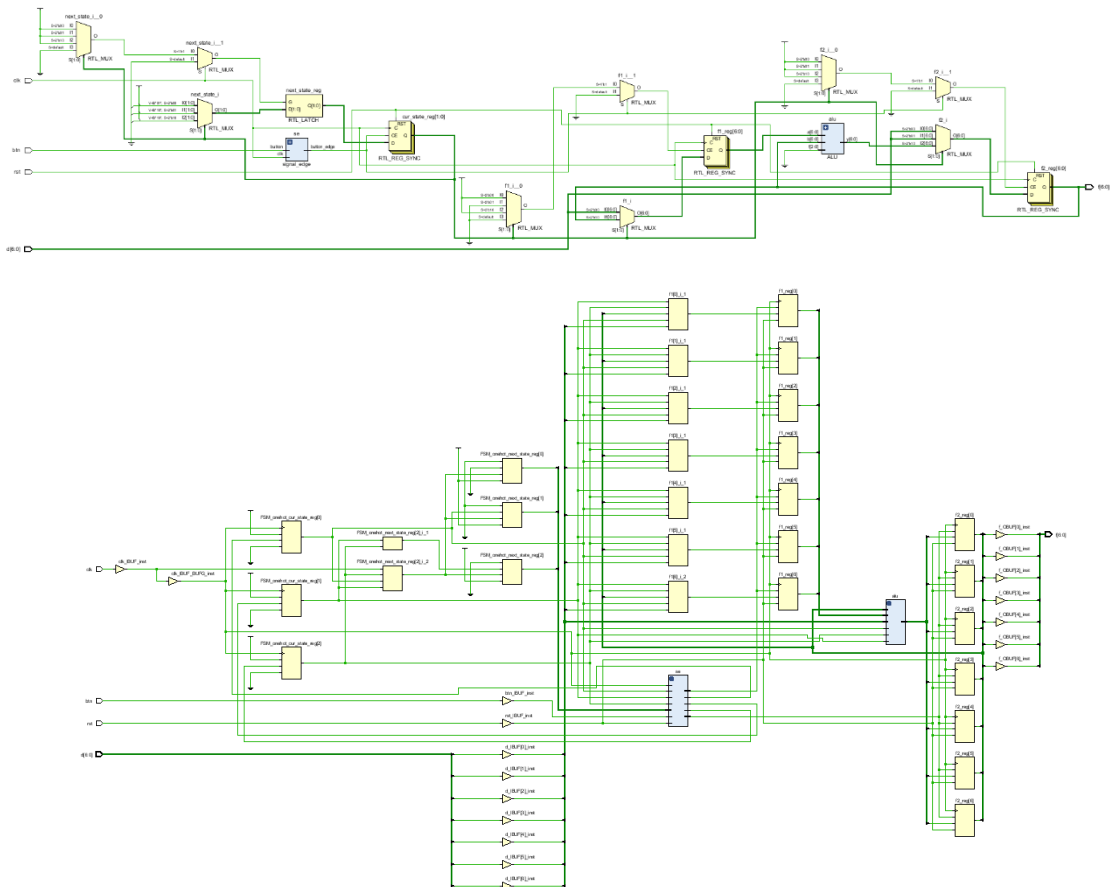
```
1 assign f = f2;
```

```

1  always @(posedge clk) begin
2      if(rst) begin
3          cur_state ≤ GET_D0;
4          f1 ≤ 7'b0;
5          f2 ≤ 7'b0;
6      end
7      else begin
8          if(en)begin
9              case (cur_state)
10                 GET_D0: begin
11                     f1 ≤ d; f2 ≤ d;
12                 end
13                 GET_D1: f2 ≤ d;
14                 CAL_F: begin
15                     f2 ≤ sum;
16                     f1 ≤ f2;
17                 end
18             endcase
19             cur_state ≤ next_state;
20         end
21     end
22 end

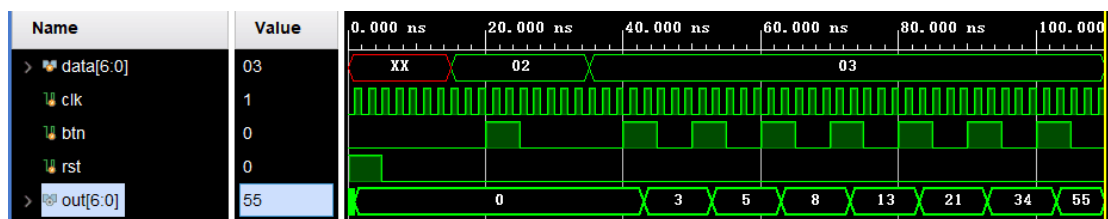
```

## 2. 电路

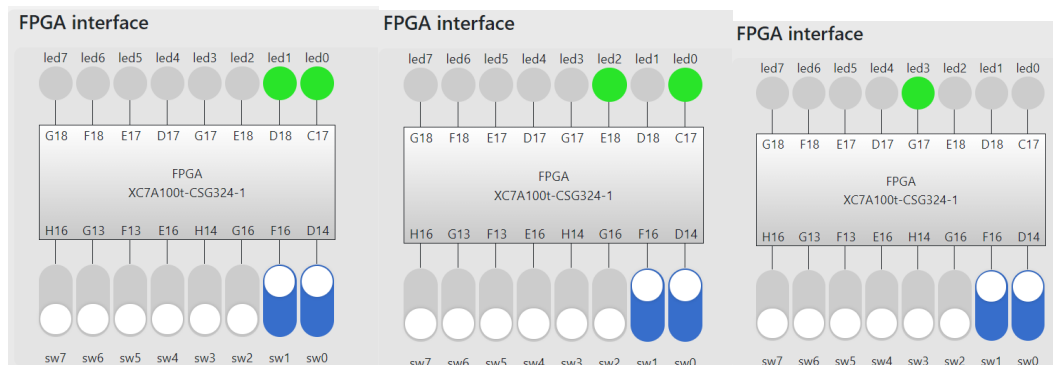


## 3. 仿真





#### 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

```

```
        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;
    end
end
```

```

        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

```

[illegible]