

# Lab 1. 运算器及其应用

姓名:傅申 学号: PB20000051 实验日期: 2022-3-15

## 1 实验题目

运算器及其应用

## 2 实验目的

- 掌握算术逻辑单元 (ALU) 的功能
- 掌握数据通路和控制器的设计方法
- 掌握组合电路和时序电路, 以及参数化和结构化的 Verilog 描述方法
- 了解查看电路性能和资源使用情况

## 3 实验平台

- Xilinx Vivado v2019.1
- Microsoft Visual Studio Code
- FPGAOOL

## 4 实验过程

### 4.1 ALU

#### 4.1.1 32 位操作数 ALU

32 位 ALU 的文件结构如下

```
main (main.v)
└── alu32: alu (alu.v)
```

ALU 核心模块的代码如下

```
1 | module alu #(parameter WIDTH = 32) // data width
2 | (
3 |     input [WIDTH - 1:0] a, b, // operands
4 |     input [2:0] f, // operation
```

```

5     output reg [WIDTH - 1:0] y,      // output
6     output z                        // zero
7 );
8     assign z = (y == {WIDTH{1'h0}});
9     always @(*) begin
10         case (f)
11             3'b000:
12                 y = a + b;
13             3'b001:
14                 y = a - b;
15             3'b010:
16                 y = a & b;
17             3'b011:
18                 y = a | b;
19             3'b100:
20                 y = a ^ b;
21             default:
22                 y = {WIDTH{1'h0}};
23         endcase
24     end
25 endmodule

```

为了查看时间性能报告, 需要在输入输出端口处加上寄存器. 用另一个 Verilog 文件实例化 ALU 模块, 并命名为 **main**, 模块如下.

```

1     module main(
2         input clk,
3         input [31:0] a, b,
4         input [2:0] f,
5         output reg [31:0] y,
6         output reg z
7     );
8         // wire mapping
9         reg [31:0] alu_a, alu_b;
10        reg [2:0] alu_f;
11        wire [31:0] alu_y;
12        wire alu_z;
13        alu #(.WIDTH(32)) alu32(
14            .a(alu_a),
15            .b(alu_b),

```

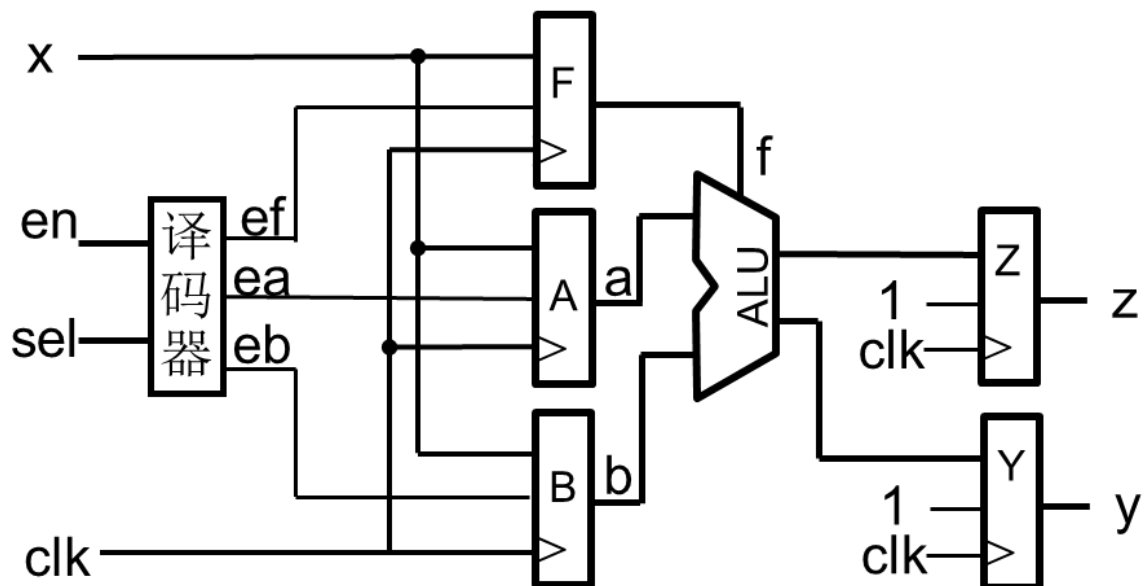
```

16         .f(alu_f),
17         .y(alu_y),
18         .z(alu_z)
19     );
20
21     // registers
22     always @(posedge clk) begin
23         alu_a <= a;
24         alu_b <= b;
25         alu_f <= f;
26         y <= alu_y;
27         z <= alu_z;
28     end
29 endmodule

```

## 4.2 6 位操作数 ALU

6 位操作数 ALU 的数据通路如下 (来自 PPT)



6 位 ALU 的设计文件结构如下

```

main (main.v)
├── dec: decoder (decoder.v)
└── alu1: alu (alu.v)

```

其中译码器模块的代码如下

```

1  module decoder(
2      input en,
3      input [1:0] sel,
4      output ea, eb, ef
5  );
6      assign ea = en & (sel == 2'b00);
7      assign eb = en & (sel == 2'b01);
8      assign ef = en & (sel == 2'b10);
9  endmodule

```

而 ALU 是通过实例化 32 位 ALU 模块实现的, 顶层模块 **main** 如下

```

1  module main #(parameter WIDTH = 6)
2  (
3      input clk,
4      input en,
5      input [1:0] sel,
6      input [WIDTH - 1:0] x,
7      output reg [WIDTH - 1:0] y,
8      output reg z
9  );
10     // wire mapping
11     wire ef, ea, eb;
12     wire alu_z;
13     wire [WIDTH - 1:0] alu_y;
14     reg [2:0] f;
15     reg [WIDTH - 1:0] a, b;
16     decoder dec(
17         .en(en),
18         .sel(sel),
19         .ef(ef),
20         .ea(ea),
21         .eb(eb)
22     );
23     alu #(.WIDTH(WIDTH)) alu1
24     (
25         .a(a),
26         .b(b),
27         .f(f),

```

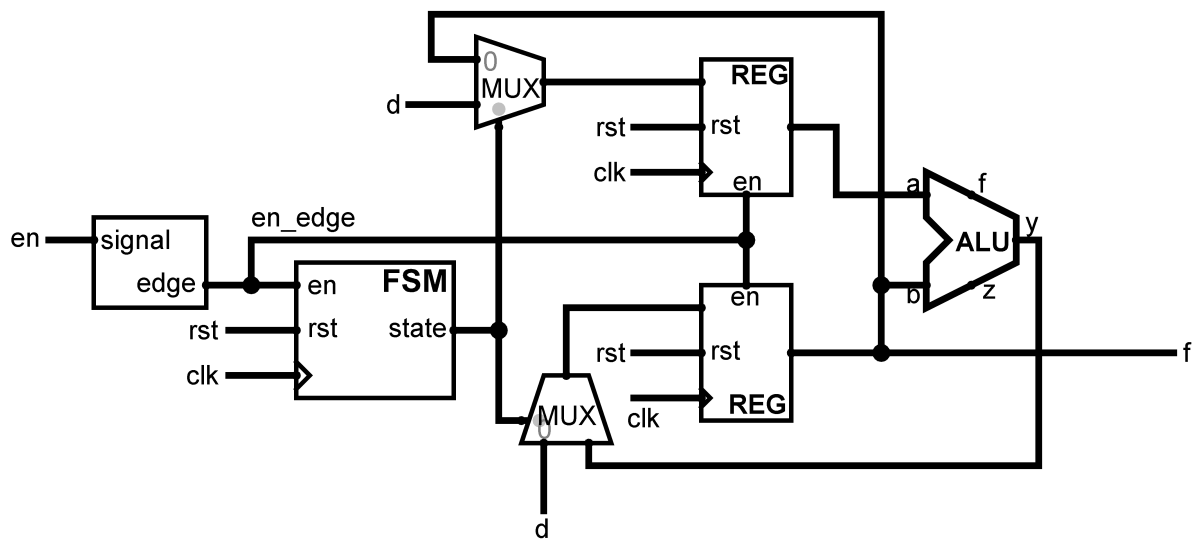
```

28         .y(alu_y),
29         .z(alu_z)
30     );
31     // registers
32     always @(posedge clk) begin
33         if (ef) f <= x[2:0];
34         if (ea) a <= x;
35         if (eb) b <= x;
36         y <= alu_y;
37         z <= alu_z;
38     end
39 endmodule

```

### 4.3 FLS

FLS 模块的数据通路如下 (MUX 的输入有所不同), 可以看到输出信号 **f** 同时作为 ALU 的一个输入.



FLS 的设计文件结构如下

```

fls (fls.v)
├── get_en_edge: get_edge (get_edge.v)
├── adder: alu (alu.v)
└── fsm1: fsm (fsm.v)

```

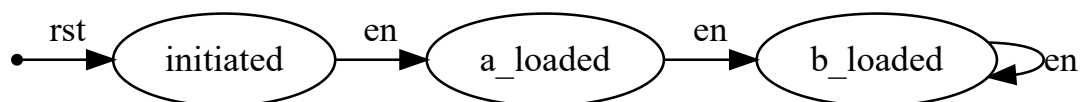
首先, 取信号边缘的模块 **get\_edge** 如下

```

1  module get_edge(
2      input clk,
3      input rst,
4      input signal,
5      output signal_edge
6  );
7      reg signal_r1, signal_r2;
8      always @(posedge clk) signal_r1 <= ~rst & signal;
9      always @(posedge clk) signal_r2 <= signal_r1;
10     assign signal_edge = signal_r1 & ~signal_r2;
11 endmodule

```

而状态机有三个状态, 分别为 **initiated**, **a\_loaded**, **b\_loaded**, 输出为当前状态, 状态转移图如下



对应的 Verilog 模块如下

```

1  module fsm(
2      input clk,
3      input rst,
4      input en,
5      output [1:0] state
6  );
7      reg [1:0] curr_state;
8      reg [1:0] next_state;
9
10     parameter initiated = 2'b00;
11     parameter a_loaded = 2'b01;
12     parameter b_loaded = 2'b10;
13
14     // FSM Part 1: state transfer
15     always @(posedge clk) begin
16         if (rst) curr_state <= initiated;
17         else if (en) curr_state <= next_state;
18     end
19

```

```

20 // FSM Part 2: next state
21 always @(curr_state) begin
22     case (curr_state)
23         initiated: next_state = a_loaded;
24         a_loaded: next_state = b_loaded;
25         b_loaded: next_state = b_loaded;
26         default: next_state = initiated;
27     endcase
28 end
29
30 // FSM Part 3: output logic and state action
31 assign state = curr_state;
32 endmodule

```

通过实例化上面两个模块以及 ALU, 最后 FLS 的 Verilog 模块如下. 其中在 FSM 状态转移时刻 (`en_edge` 为高电平), ALU 输入处的寄存器被激活, 对应的信号被传入 ALU.

```

1  module fls(
2      input clk,
3      input rst,
4      input en,
5      input [6:0] d,
6      output reg [6:0] f
7  );
8      // Wire mapping
9      // get edge
10     wire en_edge;
11     get_edge get_en_edge(
12         .clk(clk),
13         .rst(rst),
14         .signal(en),
15         .signal_edge(en_edge)
16     );
17     // ALU
18     reg [6:0] a;
19     wire [6:0] alu_out;
20     alu #(.WIDTH(7)) adder(
21         .a(a),
22         .b(f),
23         .f(3'b000),

```

```

24         .y(alu_out)
25     );
26
27     // FSM
28     wire [1:0] sel;
29     fsm fsm1(
30         .clk(clk),
31         .rst(rst),
32         .en(en_edge),
33         .state(sel)
34     );
35
36     // registers and MUXes
37     always @(posedge clk) begin
38         if (rst) a <= 7'h00;
39         else if (en_edge) begin
40             case (sel)
41                 2'b00: a <= d;
42                 2'b10: a <= f;
43                 default: a <= a;
44             endcase
45         end
46     end
47     always @(posedge clk) begin
48         if (rst) f <= 7'h00;
49         else if (en_edge) begin
50             case (sel)
51                 2'b00: f <= d;
52                 2'b01: f <= d;
53                 2'b10: f <= alu_out;
54                 default: f <= f;
55             endcase
56         end
57     end
58 endmodule

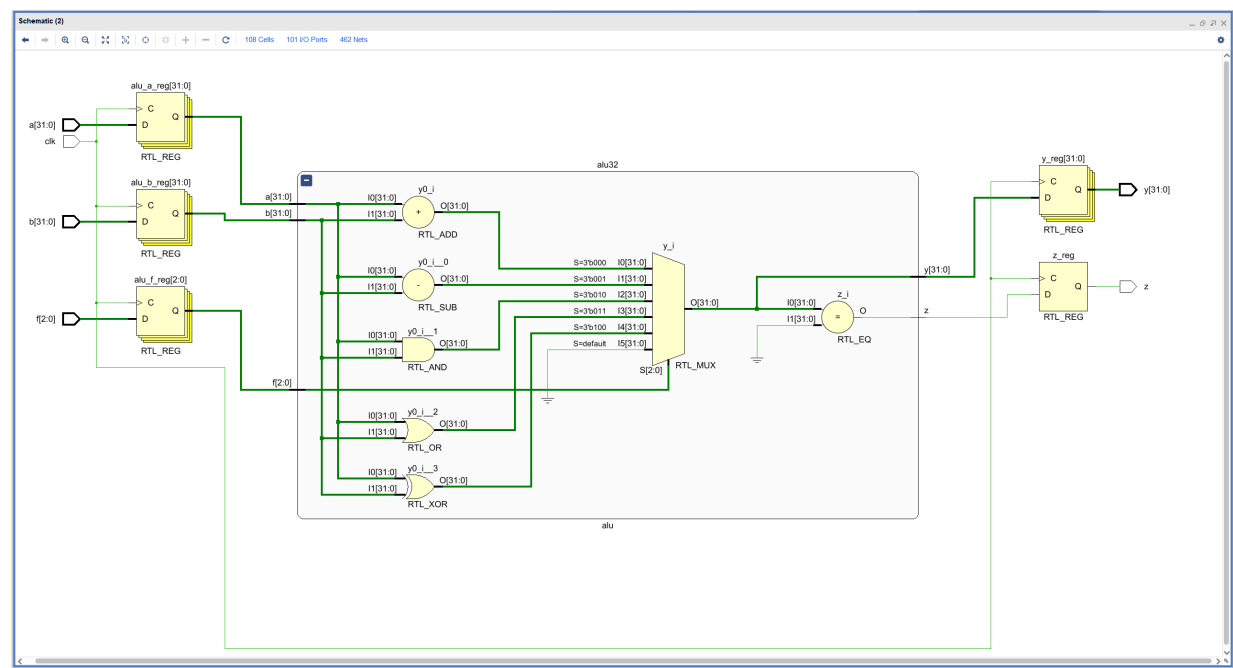
```

## 5 实验结果

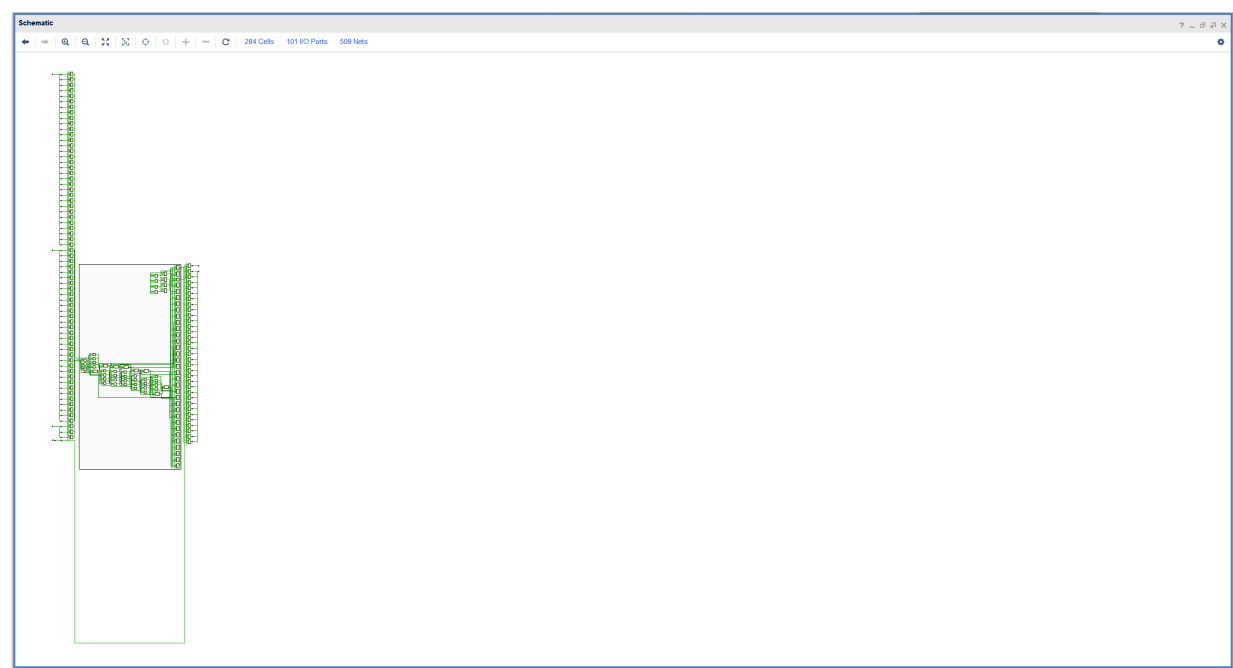
### 5.1 32 位 ALU



RTL 电路图如下



综合电路图如下



综合电路资源报告如下

Utilization					
Hierarchy					
Summary					
Hierarchy					
Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)	
main	73	100	101	1	
alu32 (alu)	73	0	0	0	
LUT as Logic (<1%)					
Register as Flip Flop (<1%)					
Memory					
DSP					
IO and GT Specific					
Bonded IOB (48%)					
IOB Master Pads					
Clocking					
BUFGCTRL (3%)					
Specific Feature					
Primitives					
Black Boxes					
Instantiated Netlists					

时间性能报告如下

Timing													
Intra-Clock Paths - sys_clk_pin - Setup													
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Clock Uncertainty
Path 1	4.298	12	13	2	alu_b_reg[1]C	z_reg[D	5.566	2.906	2.660	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 2	5.996	10	11	2	alu_b_reg[1]C	y_reg[31]D	3.868	2.646	1.222	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 3	6.073	10	11	2	alu_b_reg[1]C	y_reg[30]D	3.791	2.565	1.226	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 4	6.113	9	10	2	alu_b_reg[1]C	y_reg[27]D	3.751	2.529	1.222	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 5	6.127	10	11	2	alu_b_reg[1]C	y_reg[29]D	3.737	2.651	1.086	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 6	6.190	9	10	2	alu_b_reg[1]C	y_reg[26]D	3.674	2.448	1.226	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 7	6.230	8	9	2	alu_b_reg[1]C	y_reg[23]D	3.634	2.412	1.222	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 8	6.244	9	10	2	alu_b_reg[1]C	y_reg[25]D	3.620	2.534	1.086	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 9	6.244	10	11	2	alu_b_reg[1]C	y_reg[28]D	3.620	2.535	1.085	10.0	sys_clk_pin	sys_clk_pin	0.035
Path 10	6.307	8	9	2	alu_b_reg[1]C	y_reg[22]D	3.557	2.331	1.226	10.0	sys_clk_pin	sys_clk_pin	0.035

针对下面的仿真文件

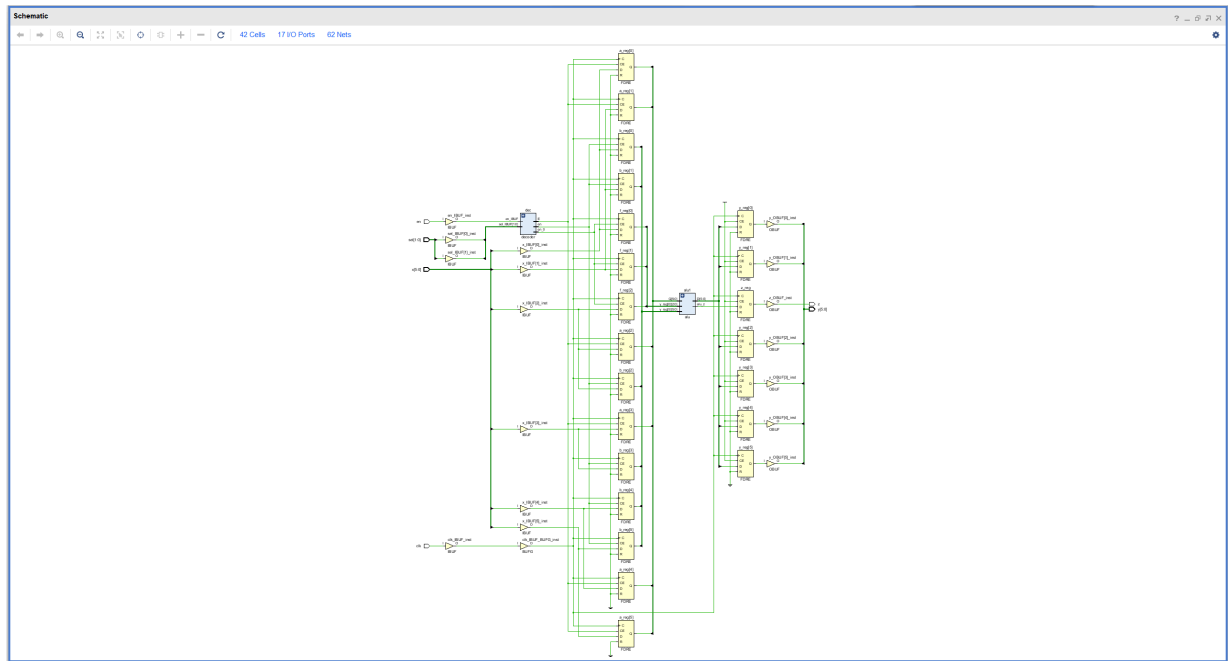
```

1  module tb();
2      parameter clk_sep = 1;
3      parameter sep = 10;
4      reg clk;
5      reg [31:0] a;
6      reg [31:0] b;
7      reg [2:0] f;
8      wire [31:0] y;
9      wire z;
10     main test(
11         .clk(clk),
12         .a(a),
13         .b(b),
14         .f(f),
15         .y(y),
16         .z(z)
17     );
18     initial begin
19         clk = 1'b0;
20         a = 32'ha5a5a5a5;
21         b = 32'h5a5aa5a5;
22         f = 3'b000;
23     end
24     always #(clk_sep) clk = ~clk;
25     initial begin
26         #(sep) f = 3'b001;
27         #(sep) f = 3'b010;
28         #(sep) f = 3'b011;
29         #(sep) f = 3'b100;
30         #(sep) f = 3'b101;
31         #(sep) f = 3'b110;
32         #(sep) f = 3'b111;
33         #(sep) $finish;
34     end
35 endmodule

```

仿真结果如下





综合电路资源报告如下

Utilization					
Hierarchy					
Name	1	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
main		15	22	17	1
alu1 (alu)		13	0	0	0
dec (decoder)		2	0	0	0

utilization\_1

时间性能报告如下

Timing

<

针对下面的仿真文件

```

1  module tb();
2      parameter timesep = 100;
3      parameter timesep_clk = 10;
4      parameter width = 6;
5
6      reg clk, en;
7      reg [1:0] sel;
8      reg [width-1:0] x;
9
10     wire [width-1:0] y;
11     wire z;
12
13     main #(.WIDTH(width)) testbench(
14         .clk(clk),
15         .en(en),
16         .sel(sel),
17         .x(x),
18         .y(y),
19         .z(z)
20     );
21     always #(timesep_clk) clk = ~clk;
22     initial begin
23         clk = 1'b0;
24         en = 1'b0;
25         sel = 2'b11;

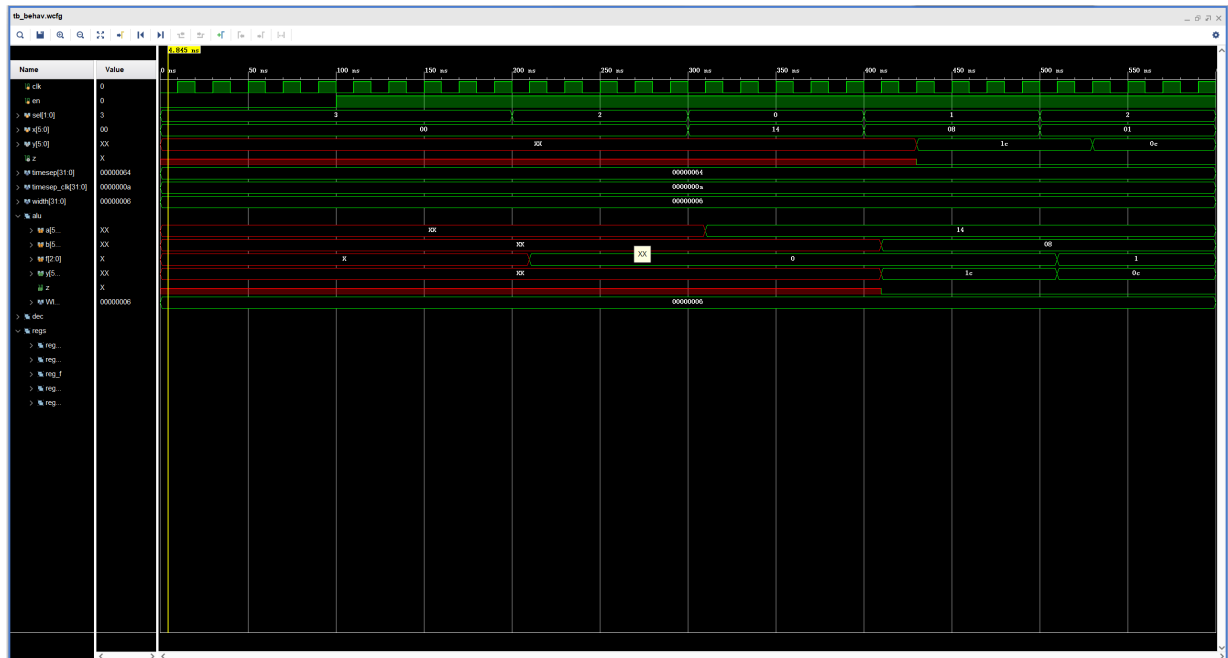
```

```

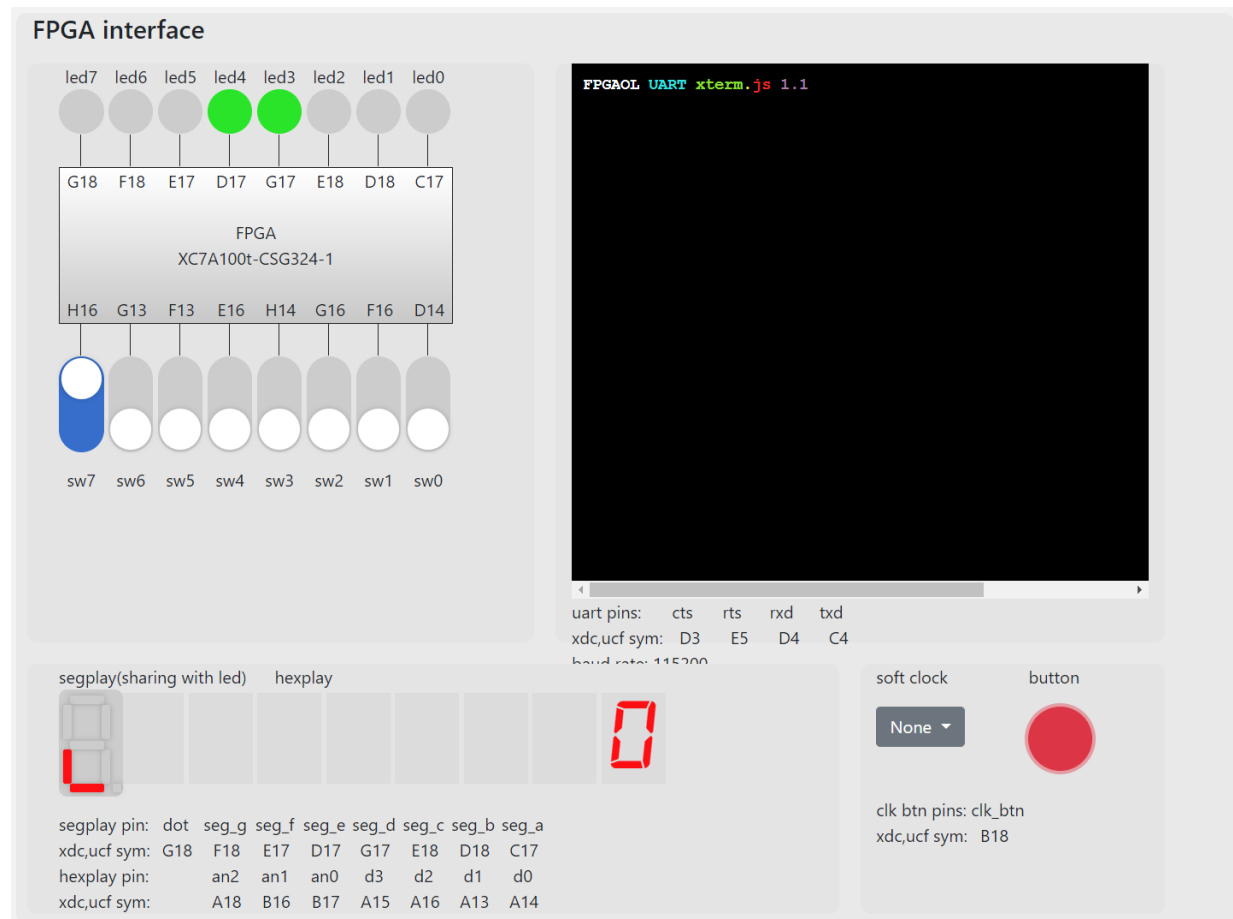
26     x = 6'h00;
27     #(timesep)
28     en = 1'b1;
29     #(timesep)
30     sel = 2'b10;
31     x = 6'h00;
32     #(timesep)
33     sel = 2'b00;
34     x = 6'h14;
35     #(timesep)
36     sel = 2'b01;
37     x = 6'h08;
38     #(timesep)
39     sel = 2'b10;
40     x = 6'h01;
41     #(timesep)
42     $finish;
43 end
44 endmodule

```

仿真结果如下



下载测试: 在 FPGAOL 上计算  $8 + 16$ , 输出如下图



### 5.3 FLS

针对下面的仿真文件

```

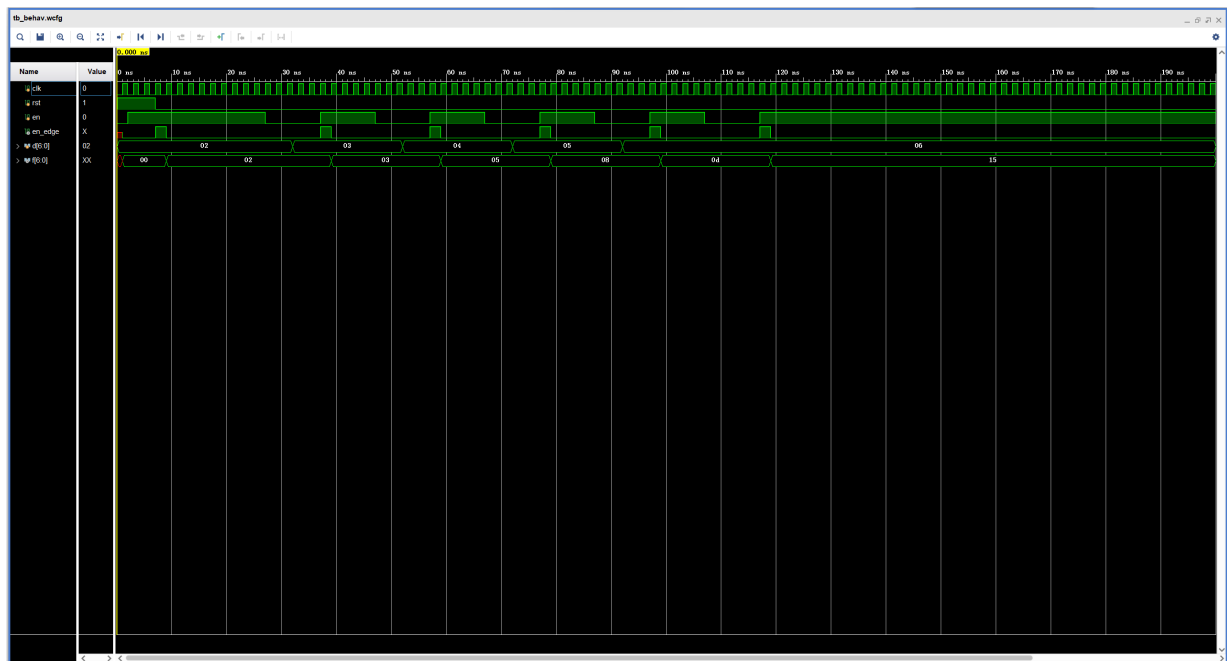
1  module tb();
2      reg clk;
3      reg rst;
4      reg en;
5      reg [6:0] d;
6      wire [6:0] f;
7
8      fls test(
9          .clk(clk),
10         .rst(rst),
11         .en(en),
12         .d(d),
13         .f(f)
14     );
15
16     parameter timesep = 1;

```



```
17
18     always #(timesep) clk = ~clk;
19
20     initial begin
21         clk = 1'b0;
22         #200 $finish;
23     end
24
25     initial begin
26         rst = 1'b1;
27         #7 rst = 1'b0;
28     end
29
30     initial begin
31         en = 1'b0;
32         #2  en = 1'b1;
33         #25 en = 1'b0;
34         #10 en = 1'b1;
35         #10 en = 1'b0;
36         #10 en = 1'b1;
37         #10 en = 1'b0;
38         #10 en = 1'b1;
39         #10 en = 1'b0;
40         #10 en = 1'b1;
41         #10 en = 1'b0;
42         #10 en = 1'b1;
43     end
44
45     initial begin
46         d = 7'h02;
47         #32 d = 7'h03;
48         #20 d = 7'h04;
49         #20 d = 7'h05;
50         #20 d = 7'h06;
51     end
52 endmodule
```

仿真结果如下



在 FPGAOL 上以 1 和 2 分别作为第一, 二个输入, 一系列输出如下

### FPGA interface

led7 led6 led5 led4 led3 led2 led1 led0

G18 F18 E17 D17 G17 E18 D18 C17

FPGA

XC7A100t-CSG324-1

H16 G13 F13 E16 H14 G16 F16 D14

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

sw0

FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd

xdc,ucf sym: D3 E5 D4 C4

baud rate: 115200

segplay(sharing with led) hexplay

segplay pin: dot seg\_g seg\_f seg\_e seg\_d seg\_c seg\_b seg\_a

xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17

hexplay pin: an2 an1 an0 d3 d2 d1 d0

xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

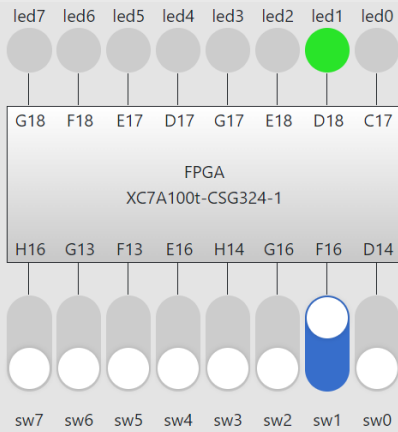
soft clock button

None

clk btn pins: clk\_btn

xdc,ucf sym: B18

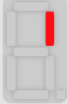
## FPGA interface



FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd  
 xdc,ucf sym: D3 E5 D4 C4  
 baud rate: 115200

segplay(sharing with led) hexplay



segplay pin: dot seg\_g seg\_f seg\_e seg\_d seg\_c seg\_b seg\_a  
 xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17  
 hexplay pin: an2 an1 an0 d3 d2 d1 d0  
 xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock

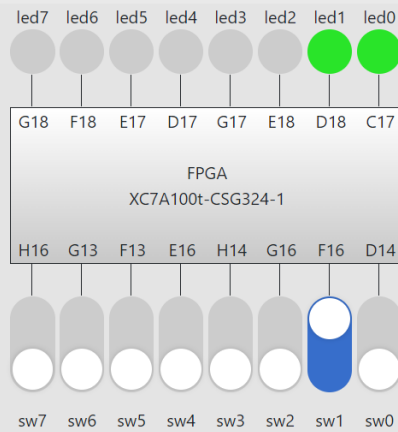
button

None



clk btn pins: clk\_btn  
 xdc,ucf sym: B18

## FPGA interface



FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd  
 xdc,ucf sym: D3 E5 D4 C4  
 baud rate: 115200

segplay(sharing with led) hexplay



segplay pin: dot seg\_g seg\_f seg\_e seg\_d seg\_c seg\_b seg\_a  
 xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17  
 hexplay pin: an2 an1 an0 d3 d2 d1 d0  
 xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock

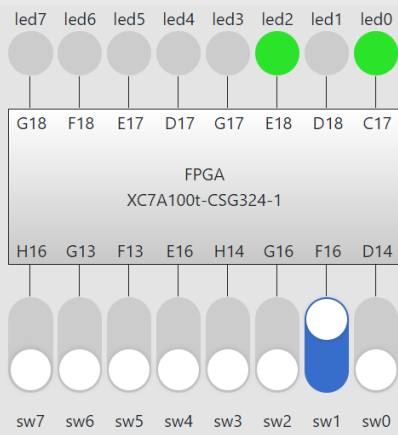
button

None



clk btn pins: clk\_btn  
 xdc,ucf sym: B18

## FPGA interface



FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd  
xdc,ucf sym: D3 E5 D4 C4  
baud rate: 115200

segplay(sharing with led) hexplay



segplay pin: dot seg\_g seg\_f seg\_e seg\_d seg\_c seg\_b seg\_a  
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17  
hexplay pin: an2 an1 an0 d3 d2 d1 d0  
xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock button

None ▾



clk btn pins: clk\_btn  
xdc,ucf sym: B18

### FPGA interface

led7 led6 led5 led4 led3 led2 led1 led0

G18 F18 E17 D17 G17 E18 D18 C17

FPGA  
XC7A100t-CSG324-1

H16 G13 F13 E16 H14 G16 F16 D14

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

```
FPGAOL UART xterm.js 1.1
```

uart pins: cts rts rxd txd  
xdc,ucf sym: D3 E5 D4 C4  
baud rate: 115200

segplay(sharing with led) hexplay

segplay pin: dot seg\_g seg\_f seg\_e seg\_d seg\_c seg\_b seg\_a  
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17  
hexplay pin: an2 an1 an0 d3 d2 d1 d0  
xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock button

None ▾

clk btn pins: clk\_btn  
xdc,ucf sym: B18

## 6 心得体会

实验难度适中.