

浙江大学



计算机逻辑设计基础

Lab 12

Author: 苏煜程

Student ID: 3220105481

Date: 2023 年 12 月 20 日

浙江大学实验报告

课程名称：计算机逻辑设计基础

实验名称：寄存器和寄存器传输设计

学生姓名：苏煜程 专业：人工智能（图灵班） 学号：3220105481

指导老师：董亚波 实验地点：东 4-509 实验日期：2023 年 12 月 7 日

1 实验目的

- 掌握寄存器传输电路的工作原理
- 掌握寄存器传输电路的设计方法
- 掌握 ALU 和寄存器传输电路的综合应用

2 实验任务

- 基于 ALU 的数据传输应用设计

3 实验原理

3.1 寄存器

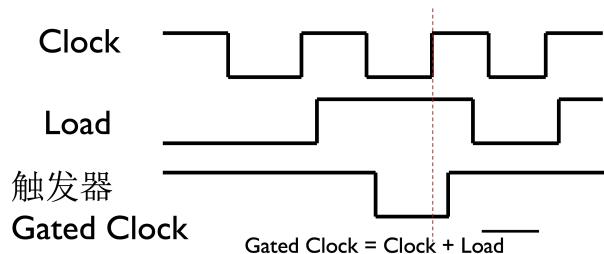
一组二进制存储单元。一个寄存器可以用于存储一列二进制值，通常用于进行简单数据存储、移动和处理等操作。

能存储信息并保存多个时钟周期，能用信号来控制“保存”或“加载”信息

3.1.1 采用门控时钟的寄存器

如果 Load 信号为 1，允许时钟信号通过，如果为 0 则阻止时钟信号通过。

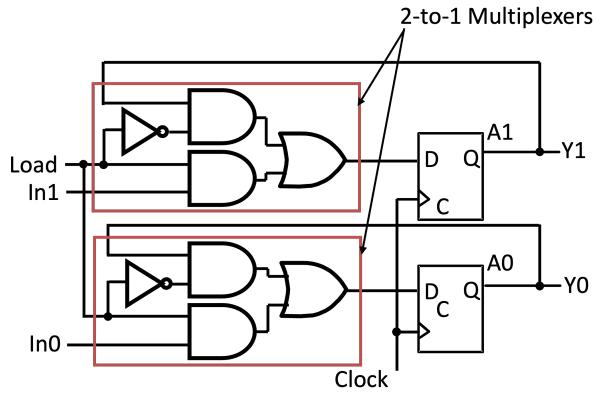
对于上升沿触发的边沿触发器或负向脉冲触发的主从触发器：



3.1.2 采用 Load 控制反馈的寄存器

进行有选择地加载寄存器的更可靠方法是：

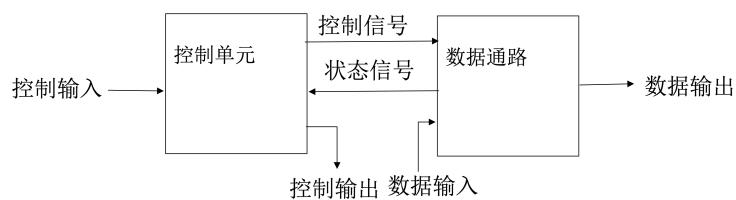
- 保证时钟的连续性
- 选择性地使用加载控制来改变寄存器的内容



```
1 ...
2 reg [3:0] OUT;
3 ...
4 always @ (posedge clk) begin
5     if (load) OUT <= IN;
6 end
7 ...
```

3.2 寄存器传输

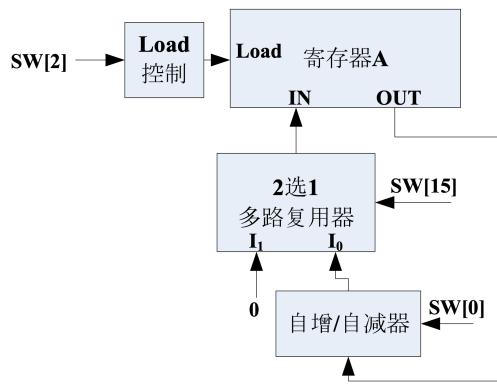
- 寄存器传输：寄存器中数据的传输和处理



- 三个基本单元：寄存器组、操作、操作控制
- 基本操作：加载、计数、移位、加法、按位操作等

3.2.1 采用寄存器传输原理的计数器

- 功能: SW[2] 拨动一次, 计一次数
- Load 控制模块——在 SW[2] 的上升沿产生 1 个时钟周期宽度的 Load 信号
- 自增/自减器可以用 4 位加减法器实现
- 功能:
 - SW[2]: 寄存器加载
 - SW[0]: 向上/下计数
 - SW[15]: 寄存器清零

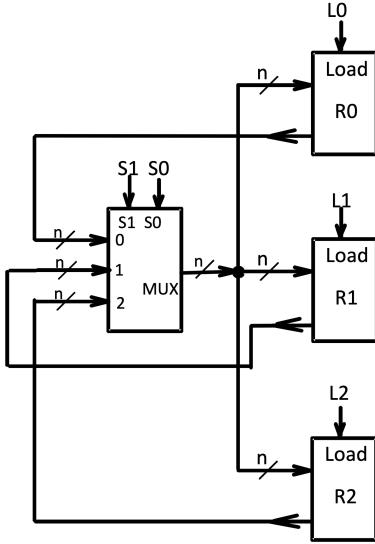


其中 Load 控制模块实现如下:

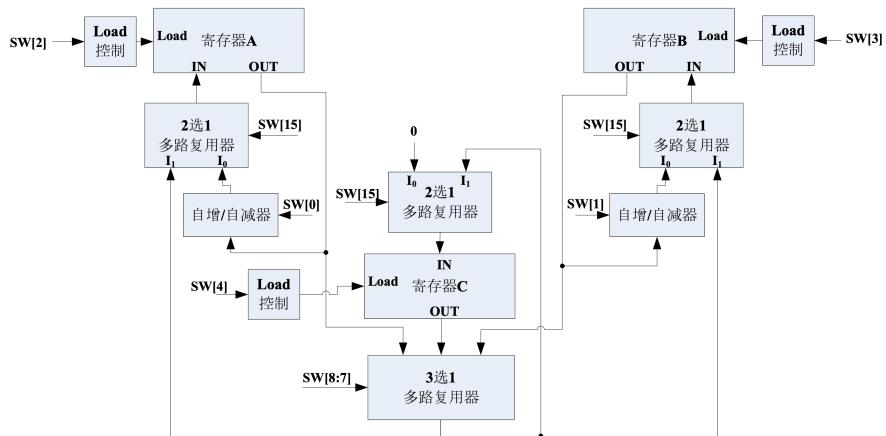
```
1 module Load_Gen(
2     input wire clk,
3     input wire clk_1ms,
4     input wire btn_in,
5     output reg Load_out
6 );
7     initial Load_out = 0;
8     wire btn_out;
9     reg old_btn;
10    pbdebounce p0(clk_1ms, btn_in, btn_out);
11    always@(posedge clk) begin
12        if ((old_btn == 1'b0) && (btn_out == 1'b1)) // btn 出现上升沿
13            Load_out <= 1'b1;
14        else
15            Load_out <= 1'b0;
16    end
17    always@(posedge clk) begin // 保存上一个周期 btn 的状态
18        old_btn <= btn_out;
19    end
20 endmodule
```

3.3 基于多路选择器总线的寄存器传输

由一个多路选择器驱动的总线可以降低硬件开销，这个结构不能实现多个寄存器相互之间的并行传输操作。



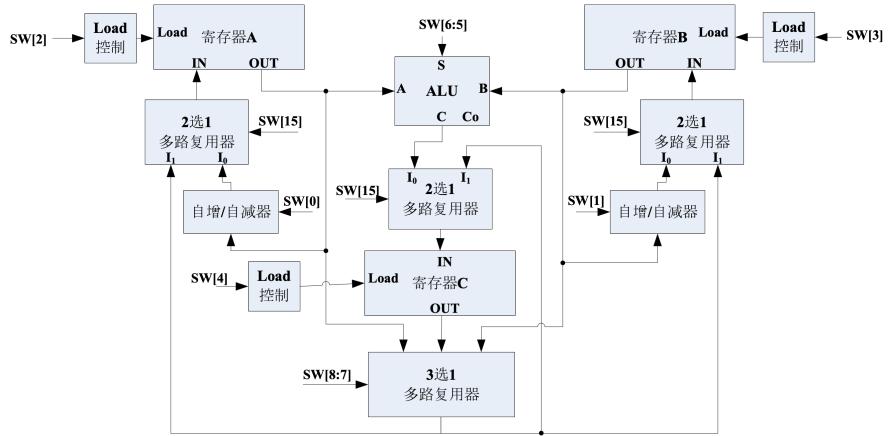
- SW[2]、SW[3]、SW[4]: 更新 A、B、C 寄存器，更新的值由 SW[15] 和 SW[0]、SW[1] 决定
- SW[15]=0: 通过向上/向下计数修改寄存器 A、B，对 C 清零
- SW[15]=1: A、B、C 寄存器之间相互传输，源寄存器由 SW[8:7] 确定



3.4 寄存器传输应用设计

加入 ALU 模块：

- 功能：在上一个任务基础上，可以用 A 与 B 的 ALU 计算结果初始化 C
- SW[15]=0: 初始化寄存器 A、B，ALU 运算输出修改继寄存器 C



- SW[15]=1: A、B、C 寄存器之间相互传输

4 实验内容与步骤

4.1 任务 1：采用寄存器传输原理设计计数器

设计任务 1 的 Top Module 主要代码：

```

1 `timescale 1ns / 1ps
2 module top(
3     input wire clk,
4     input wire [15:0] SW,
5     output wire [3:0] AN,
6     output wire [7:0] Segment
7 );
8
9     wire Load;
10    wire [3:0] A_OUT, A_IN, A1;
11    wire [31:0] clk_div;
12
13    MyRegister4b m0(.clk(clk),
14                      .IN(A_IN),
15                      .Load(Load),
16                      .OUT(A_OUT));
17
18    Load_Gen m1(.clk(clk),
19                  .clk_1ms(clk_div[17]),
20                  .btn_in(SW[2]),
21                  .Load_out(Load));
22
23    clkdiv m2(.clk(clk),
24                  .rst(1'b0),
25                  .clkdiv(clk_div));
26

```

```

27     AddSub4b m3(.A(A_OUT),
28                 .B(4'b1),
29                 .Ctrl(SW[0]),
30                 .S(A1));
31
32     assign A_IN = (SW[15] == 1'b0) ? A1 : 4'b0;
33
34     DispNum m4(.clk(clk),
35                 .HEXS({A_OUT, A1, A_IN, 4'b0}),
36                 .LES(4'b0),
37                 .point(4'b0),
38                 .RST(1'b0),
39                 .AN(AN),
40                 .Segment(Segment));
41
endmodule

```

4.1.1 仿真实验

对 Load_Gen 模块和 top.v 作相应修改：

```

1 `timescale 1ns / 1ps
2 module Load_Gen(
3     input wire clk,
4     input wire clk_1ms,
5     input wire btn_in,
6     output reg Load_out
7 );
8     initial Load_out = 0;
9 // wire btn_out;
10    reg old_btn;
11 // pbdebounce p0(clk_1ms, btn_in, btn_out);
12    always @ (posedge clk) begin
13        if ((old_btn == 1'b0) && (btn_in == 1'b1)) Load_out <= 1'b1;
14        else Load_out <= 1'b0;
15    end
16    always @ (posedge clk) begin
17        old_btn <= btn_in;
18    end
19 endmodule

```

```

1 `timescale 1ns / 1ps
2 module top(
3     input wire clk,
4     input wire [15:0] SW,
5     output wire [15:0] num
6 // output wire [3:0] AN,

```

```

7      // output wire [7:0] Segment
8  );
9  wire Load;
10 wire [3:0] A_OUT, A_IN, A1;
11 wire [31:0] clk_div;
12
13 assign num = {A_OUT, A1, A_IN, 4'b0};
14
15 MyRegister4b m0(.clk(clk),
16                 .IN(A_IN),
17                 .Load(Load),
18                 .OUT(A_OUT));
19
20 Load_Gen m1(.clk(clk),
21             .clk_1ms(clk_div[17]),
22             .btn_in(SW[2]),
23             .Load_out(Load));
24
25 clkdiv m2(.clk(clk),
26            .rst(1'b0),
27            .clkdiv(clk_div));
28
29 AddSub4b m3(.A(A_OUT),
30             .B(4'b1),
31             .Ctrl(SW[0]),
32             .S(A1));
33
34 assign A_IN = (SW[15] == 1'b0) ? A1 : 4'b0;
35
36 // DispNum m4(.clk(clk),
37 //               .HEXS({A_OUT, A1, A_IN, 4'b0}),
38 //               .LES(4'b0),
39 //               .point(4'b0),
40 //               .RST(1'b0),
41 //               .AN(AN),
42 //               .Segment(Segment));
43
endmodule

```

进行仿真。仿真激励代码：

```

1 integer i;
2 initial begin
3     SW = 0;
4
5     SW[15] = 1; #20;
6     SW[2] = 1; #20;
7     SW[2] = 0; #20;

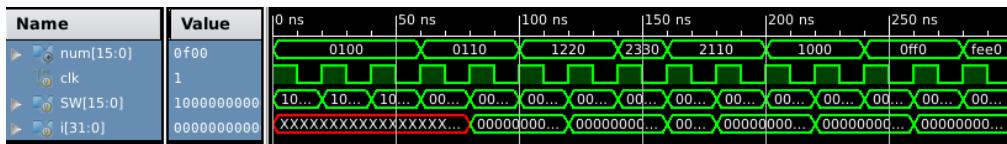
```

```

8
9     SW[15] = 0; #20;
10    for (i = 0; i < 2; i = i + 1) begin
11        SW[2] = 1; #20;
12        SW[2] = 0; #20;
13    end
14    SW[0] = 1; #20;
15    for (i = 0; i < 3; i = i + 1) begin
16        SW[2] = 1; #20;
17        SW[2] = 0; #20;
18    end
19
20    SW[15] = 1; #20;
21    SW[2] = 1; #20;
22    SW[2] = 0; #20;
23
24 end
25
26 always begin
27     clk = 1; #10;
28     clk = 0; #10;
29 end

```

仿真结果如下：



仿真代码中先将寄存器清零，对应仿真波形中 A_OUT 对应位清零。然后 SW[15] 拨到 0，SW[0] 拨到 0，SW[2] 拨动两次，寄存器存储的值加 2。然后 SW[0] 拨到 1，SW[2] 拨动三次，寄存器存储的值减 3。可见最终寄存器中的值为 F。符合预期。

4.2 任务 2：基于多路选择器总线的寄存器传输

设计任务 2 的 Top Module 主要代码：

```

1 `timescale 1ns / 1ps
2 module top(
3     input clk,
4     input [15:0] SW,
5     output [3:0] AN,
6     output [7:0] Segment
7 );
8

```

```

9      wire Load_A, Load_B, Load_C;
10     wire [3:0] A_IN, A_IO, A_OUT;
11     wire [3:0] B_IN, B_IO, B_OUT;
12     wire [3:0] C_IN, C_IO, C_OUT;
13     wire [3:0] I1;
14     wire [31:0] clk_div;
15
16     clkdiv m0(clk, 1'b0, clk_div);
17
18     MyRegister4b m1(.clk(clk),
19                      .IN(A_IN),
20                      .Load(Load_A),
21                      .OUT(A_OUT));
22
23     Load_Gen m2(.clk(clk),
24                  .clk_1ms(clk_div[17]),
25                  .btn_in(SW[2]),
26                  .Load_out(Load_A));
27
28     AddSub4b m3(.A(A_OUT),
29                  .B(4'b1),
30                  .Ctrl(SW[0]),
31                  .S(A_IO));
32
33     assign A_IN = (SW[15] == 1'b0) ? A_IO : I1;
34
35     MyRegister4b m4(.clk(clk),
36                      .IN(B_IN),
37                      .Load(Load_B),
38                      .OUT(B_OUT));
39
40     Load_Gen m5(.clk(clk),
41                  .clk_1ms(clk_div[17]),
42                  .btn_in(SW[3]),
43                  .Load_out(Load_B));
44
45     AddSub4b m6(.A(B_OUT),
46                  .B(4'b1),
47                  .Ctrl(SW[1]),
48                  .S(B_IO));
49
50     assign B_IN = (SW[15] == 1'b0) ? B_IO : I1;
51
52     MyRegister4b m7(.clk(clk),
53                      .IN(C_IN),
54                      .Load(Load_C),
55                      .OUT(C_OUT));
56
57     Load_Gen m8(.clk(clk),
58                  .clk_1ms(clk_div[17]),
59                  .btn_in(SW[4]),
60                  .Load_out(Load_C));
61
62     assign C_IN = (SW[15] == 1'b0) ? 4'b0 : I1;

```

```

56     Mux4to1b4 m9(.I0(A_OUT),
57                 .I1(B_OUT),
58                 .I2(C_OUT),
59                 .I3(4'b0),
60                 .S(SW[8:7]),
61                 .o(I1));
62
63     DispNum m10(.clk(clk),
64                 .HEXS({A_OUT, B_OUT, C_OUT, 4'b0}),
65                 .LES(4'b0),
66                 .point(4'b0),
67                 .RST(1'b0),
68                 .AN(AN),
69                 .Segment(Segment));
70
endmodule

```

4.3 任务 3：基于 ALU 的数据传输应用设计

设计任务 3 的 Top Module 主要代码：

```

1 `timescale 1ns / 1ps
2 module top(
3   input clk,
4   input [15:0] SW,
5   output [3:0] AN,
6   output [7:0] Segment
7 );
8
9   wire Load_A, Load_B, Load_C, carry;
10  wire [3:0] A_IN, A_IO, A_OUT;
11  wire [3:0] B_IN, B_IO, B_OUT;
12  wire [3:0] C_IN, C_IO, C_OUT;
13  wire [3:0] I1, IO;
14  wire [31:0] clk_div;
15
16  clkdiv m0(clk, 1'b0, clk_div);
17
18  MyRegister4b m1(.clk(clk),
19                  .IN(A_IN),
20                  .Load(Load_A),
21                  .OUT(A_OUT));
22  Load_Gen m2(.clk(clk),
23               .clk_1ms(clk_div[17]),
24               .btn_in(SW[2]),
25               .Load_out(Load_A));
26  AddSub4b m3(.A(A_OUT),

```

```

27      .B(4'b1),
28      .Ctrl(SW[0]),
29      .S(A_IO));
30
31      assign A_IN = (SW[15] == 1'b0) ? A_IO : I1;
32
33      MyRegister4b m4(.clk(clk),
34                      .IN(B_IN),
35                      .Load(Load_B),
36                      .OUT(B_OUT));
37
38      Load_Gen m5(.clk(clk),
39                    .clk_1ms(clk_div[17]),
40                    .btn_in(SW[3]),
41                    .Load_out(Load_B));
42
43      AddSub4b m6(.A(B_OUT),
44                    .B(4'b1),
45                    .Ctrl(SW[1]),
46                    .S(B_IO));
47
48      assign B_IN = (SW[15] == 1'b0) ? B_IO : I1;
49
50
51      MyRegister4b m7(.clk(clk),
52                      .IN(C_IN),
53                      .Load(Load_C),
54                      .OUT(C_OUT));
55
56      Load_Gen m8(.clk(clk),
57                    .clk_1ms(clk_div[17]),
58                    .btn_in(SW[4]),
59                    .Load_out(Load_C));
60
61      assign C_IN = (SW[15] == 1'b0) ? IO : I1;
62
63
64      myALU m9(.S(SW[6:5]),
65                  .A(A_OUT),
66                  .B(B_OUT),
67                  .O(IO),
68                  .Co(carry));
69
70      Mux4to1b4 m10(.IO(A_OUT),
71                      .I1(B_OUT),
72                      .I2(C_OUT),
73                      .I3(4'b0),
74                      .s(SW[8:7]),
75                      .o(I1));
76
77
78      DispNum m11(.clk(clk),
79                  .HEXS({A_OUT, B_OUT, C_OUT, I1}),
80                  .LES(4'b0),
81                  .point(4'b0),
82                  .RST(1'b0),
83                  .AN(AN),
84
```

```
74     .Segment(Segment));  
75  
endmodule
```

4.3.1 仿真实验

对 Load_Gen 模块和 top.v 作相应修改。

其中 Load_Gen 模块修改和任务 1 相同， top.v 修改如下：

```
1 `timescale 1ns / 1ps  
2 module top(  
3     input clk,  
4     input [15:0] SW,  
5     output wire [15:0] num  
6     // output [3:0] AN,  
7     // output [7:0] Segment  
8 );  
9  
10    wire Load_A, Load_B, Load_C, carry;  
11    wire [3:0] A_IN, A_IO, A_OUT;  
12    wire [3:0] B_IN, B_IO, B_OUT;  
13    wire [3:0] C_IN, C_IO, C_OUT;  
14    wire [3:0] I1, IO;  
15    wire [31:0] clk_div;  
16  
17    assign num = {A_OUT, B_OUT, C_OUT, I1};  
18  
19    clkdiv m0(clk, 1'b0, clk_div);  
20  
21    MyRegister4b m1(.clk(clk),  
22                    .IN(A_IN),  
23                    .Load(Load_A),  
24                    .OUT(A_OUT));  
25    Load_Gen m2(.clk(clk),  
26                  .clk_1ms(clk_div[17]),  
27                  .btn_in(SW[2]),  
28                  .Load_out(Load_A));  
29    AddSub4b m3(.A(A_OUT),  
30                  .B(4'b1),  
31                  .Ctrl(SW[0]),  
32                  .S(A_IO));  
33    assign A_IN = (SW[15] == 1'b0) ? A_IO : I1;  
34  
35    MyRegister4b m4(.clk(clk),  
36                    .IN(B_IN),  
37                    .Load(Load_B),  
38                    .OUT(B_OUT));
```

```

39      Load_Gen m5(.clk(clk),
40                  .clk_1ms(clk_div[17]),
41                  .btn_in(SW[3]),
42                  .Load_out(Load_B));
43      AddSub4b m6(.A(B_OUT),
44                  .B(4'b1),
45                  .Ctrl(SW[1]),
46                  .S(B_IO));
47      assign B_IN = (SW[15] == 1'b0) ? B_IO : I1;
48
49      MyRegister4b m7(.clk(clk),
50                      .IN(C_IN),
51                      .Load(Load_C),
52                      .OUT(C_OUT));
53      Load_Gen m8(.clk(clk),
54                  .clk_1ms(clk_div[17]),
55                  .btn_in(SW[4]),
56                  .Load_out(Load_C));
57      assign C_IN = (SW[15] == 1'b0) ? IO : I1;
58
59      myALU m9(.S(SW[6:5]),
60                  .A(A_OUT),
61                  .B(B_OUT),
62                  .O(IO),
63                  .Co(carry));
64      Mux4to1b4 m10(.IO(A_OUT),
65                      .I1(B_OUT),
66                      .I2(C_OUT),
67                      .I3(4'b0),
68                      .s(SW[8:7]),
69                      .o(I1));
70
71      // DispNum m11(.clk(clk),
72      //                 .HEXS({A_OUT, B_OUT, C_OUT, I1}),
73      //                 .LES(4'b0),
74      //                 .point(4'b0),
75      //                 .RST(1'b0),
76      //                 .AN(AN),
77      //                 .Segment(Segment));
78 endmodule

```

进行仿真。仿真激励代码：

```

1 initial begin
2     // Initialize Inputs
3     SW = 0;
4

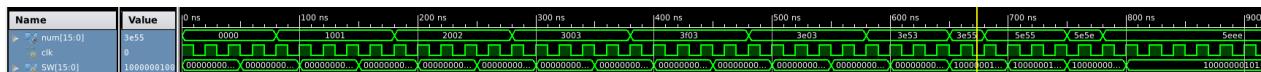
```

```

5      #50;
6
7      SW[15] = 0;
8      SW[0] = 0;
9      SW[2] = 1; #50;
10     SW[2] = 0; #50;
11     SW[2] = 1; #50;
12     SW[2] = 0; #50;
13     SW[2] = 1; #50;
14     SW[2] = 0; #50;
15
16     SW[1] = 1;
17     SW[3] = 1; #50;
18     SW[3] = 0; #50;
19     SW[3] = 1; #50;
20     SW[3] = 0; #50;
21
22     SW[6:5] = 2'b01;
23     SW[4] = 1; #50;
24     SW[4] = 0; #50;
25
26     SW[15] = 1;
27     SW[8:7] = 2'b10;
28     SW[2] = 1; #50;
29     SW[2] = 0; #50;
30
31     SW[8:7] = 2'b01;
32     SW[4] = 1; #50;
33     SW[4] = 0; #50;
34 end
35
36 always begin
37     clk = 1; #10;
38     clk = 0; #10;
39 end

```

仿真结果如下：



拨动 SW[15] 到 0，拨动 SW[0] 到 0，拨动 SW[2] 3 次，寄存器 A 加 3。拨动 SW[1] 到 1，拨动 SW[3] 2 次，寄存器 B 减 2。拨动 SW[6:5] 到 01，使得 ALU 计算结果为 A - B，拨动 SW[4] 将 ALU 计算结果写入寄存器 C。

将 SW[15] 拨动到 1，拨动 SW[8:7] 到 10，拨动 SW[2] 1 次，寄存器 C 的值传输到寄存器 A。拨动 SW[8:7] 到 01，拨动 SW[4] 1 次，寄存器 B 的值传输到寄存器 C。

可见最终寄存器 A 的值为 5，寄存器 B 的值为 E，寄存器 C 的值为 E。符合预期。

4.3.2 上板验证

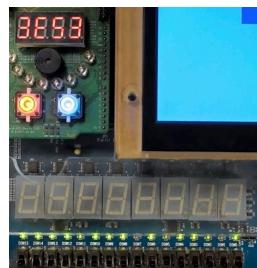
将上一节的代码改回原本的代码，生成 bit 文件，进行上板验证。验证结果如下：



(a) 拨动 SW[2] 使 A 加 3



(b) 拨动 SW[3] 使 B 减 2



(c) 将 ALU 计算结果写入 C



(d) 将 C 的值传输到 A



(e) 将 B 的值传输到 C

可见上板结果符合预期。

5 实验结果分析

实验结果在前一节已经分析，符合实验要求。

6 讨论与心得

这次实验三个任务比较有梯度，所以比较顺利，没有遇到什么问题。最近几次实验基本都是通过 verilog 实现功能，对编写代码的要求比较高，锻炼了自己的代码能力。