

浙江大学



计算机逻辑设计基础

Lab 8 & 9

Author: 苏煜程

Student ID: 3220105481

Date: 2023 年 11 月 23 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：加法器、加减法器和 ALU 基本原理与设计

学生姓名：苏煜程 专业：人工智能（图灵班） 学号：3220105481

同组学生姓名：张延泽 指导老师：董亚波

实验地点：东 4-509 实验日期：2023 年 11 月 9 日

1 实验目的

1. 掌握一位全加器的工作原理和逻辑功能
2. 掌握串行进位加法器的工作原理和进位延迟
3. 掌握减法器的实现原理
4. 掌握加减法器的设计方法
5. 掌握 ALU 基本原理及在 CPU 中的作用
6. 掌握 ALU 的设计方法

2 实验任务

- 原理图方式设计 4 位加减法器
- 实现 4 位 ALU 及应用设计

3 实验原理

3.1 一位全加器

- 三个输入位：数据位 A_i 和 B_i ，低位进位输入 C_i
- 二个输出位：全加和 S_i ，进位输出 C_{i+1}

S_i, C_{i+1} 可以用 A_i, B_i, C_i 的逻辑表达式表示：

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

真值表如下：

A_i	B_i	C_i	S_i	$C_i + 1$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

根据一位全加器的输入输出关系，得到电路图：

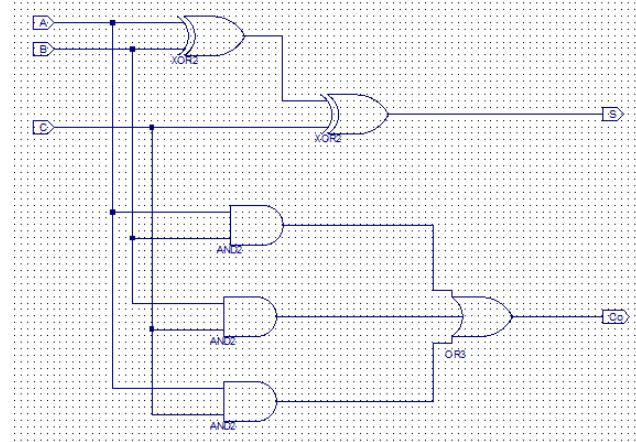


图 1: 一位全加器电路图

3.2 多位串行进位加法器

- 由一位全加器将进位串接构成
- 低位进位 C_0 为 0， C_i 为高位进位输出

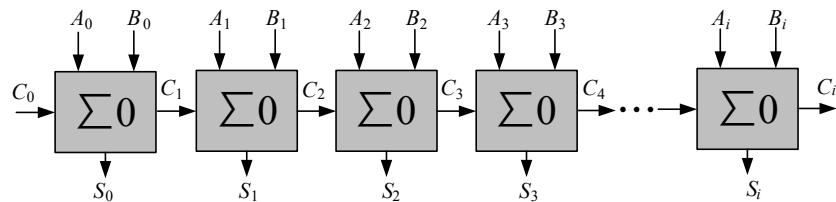


图 2: 多位串行进位加法器

由此得到四位全加器电路图：

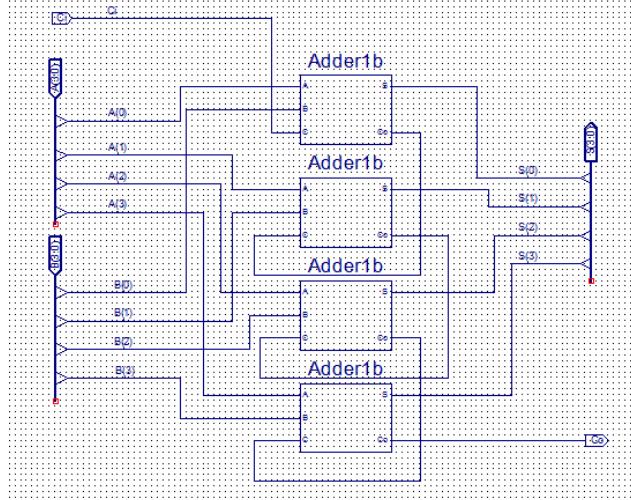


图 3: 四位全加器电路图

3.3 一位加减法器

- 用负数补码加法实现，减数当作负数求补码
- 共用加法器
- 用“异或”门控制求反，低位进位 C_0 为 1

电路图：

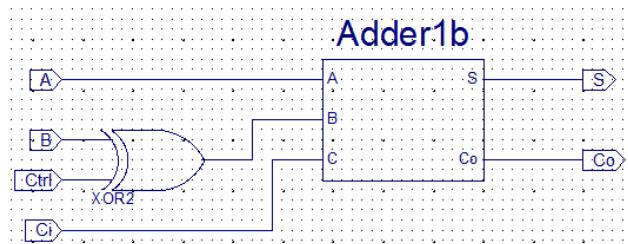


图 4: 一位加减法器电路图

3.4 多位串行进位全减器

- 用负数补码加法实现，减数当作负数求补码
- 共用加法器
- 用“异或”门控制求反，低位进位 C_0 为 1

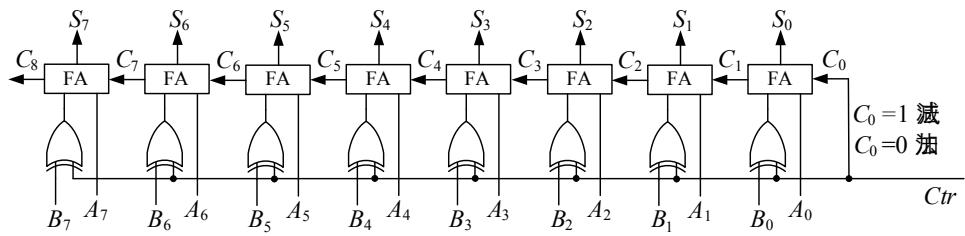


图 5: 多位串行进位全减器

由此得到四位加减法器电路图：

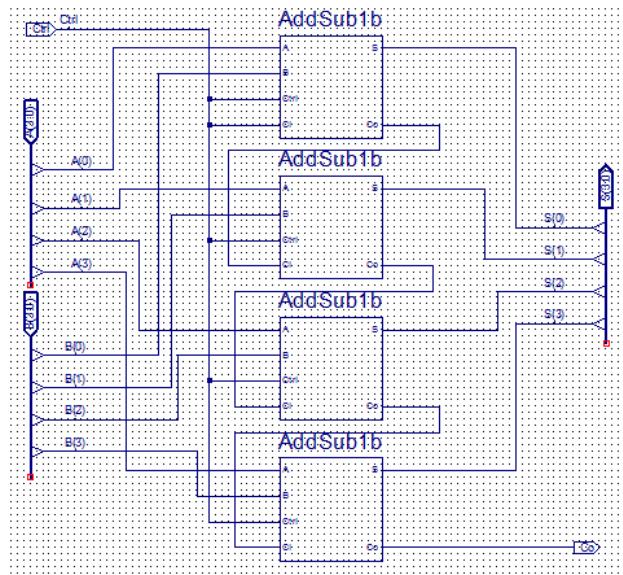


图 6: 四位加减法器电路图

3.5 四位 ALU 功能定义

- 两个 4 位操作数 $A(3:0)$, $B(3:0)$
- $S(1:0)$ 是 ALU 的功能选择引脚, 分别选择选择加、减、与、或操作
 - $S(1:0) = 00: C = A + B$
 - $S(1:0) = 01: C = A - B$
 - $S(1:0) = 10: C = A \text{ and } B$
 - $S(1:0) = 11: C = A \text{ or } B$
- ALU 计算得到进位 C_o 和结果 $C(3:0)$
- myAnd2b4、myOr2b4 分别是 4 位 2 输入与门和 4 位 2 输入或门

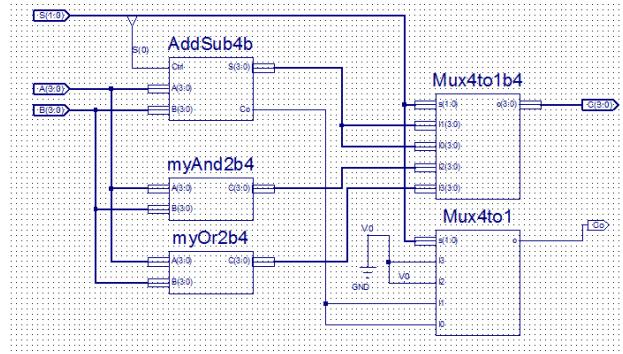


图 7: 四位 ALU 电路图

3.6 按键数据输入模块

在实验 7 基础上，更新 CreateNumber 模块

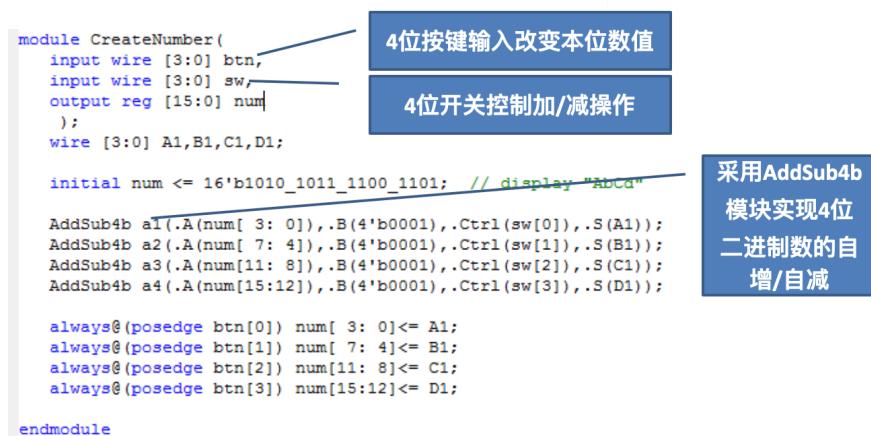


图 8: CreateNumber 模块

3.7 按键去抖动原理

- 抖动原因：按键按下或放开时，存在机械震动
- 抖动时间一般在 $10 \sim 20$ ms
- 按键去抖动方法：延时一段时间后再监测一次，以避开机械抖动



防抖动模块：

```

1 module pbdebounce(
2     input wire clk_1ms,
3     input wire button,
4     output reg pbreg
5 );
6
7     reg [7:0] pbshift;
8
9     always@(posedge clk_1ms) begin
10        pbshift=pbshift<<1;
11        pbshift[0]=button;
12        if (pbshift==8'b0)
13            pbreg=0;
14        if (pbshift==8'hFF)
15            pbreg=1;
16    end
17 endmodule

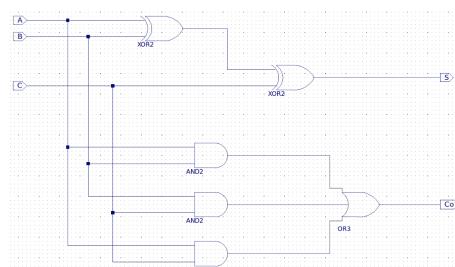
```

4 实验内容与步骤

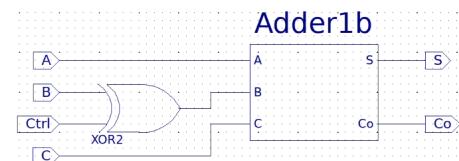
4.1 任务 1：原理图方式设计 4 位加减法器

4.1.1 一位全加器

1. 新建工程 MyAdder，Top Level Source Type 选择 Schametic
2. 新建源文件，类型是 Schametic，文件名称用 Adder1b，原理图方式进行设计
3. 新建源文件，类型是 Schametic，文件名称用 AddSub1b，原理图方式进行设计



(a) Adder1b 原理图



(b) AddSub1b 原理图

4. 进行波形仿真

```

1 integer i;
2 initial begin

```

```

3   Ctrl = 0;
4   for (i = 0; i < 8; i = i + 1) begin
5     {A, B, C} = i; #50;
6   end
7   Ctrl = 1;
8   for (i = 0; i < 8; i = i + 1) begin
9     {A, B, C} = i; #50;
10  end
11  {A, B, C} = 3'b0;
12  Ctrl = 0;
13 end

```

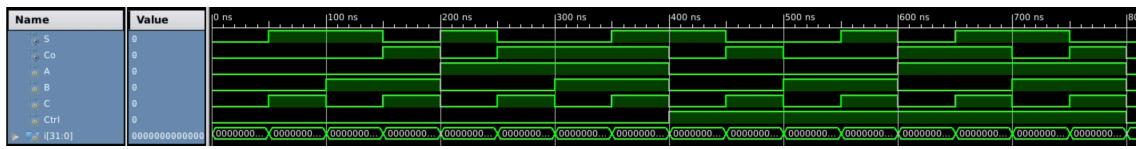


图 11: AddSub1b 波形仿真

其中当 $A = 1, B = 1, C = 0, Ctrl = 0$ 进行加法运算时, $S = 0, Co = 1$, 符合预期。

当 $A = 1, B = 1, C = 1, Ctrl = 0$ 进行加法运算时, $S = 1, Co = 1$, 符合预期。

当 $A = 0, B = 0, C = 0, Ctrl = 1$ 进行减法运算时, 此时代表存在借位, 此位不够借位, 继续向后一位借位, $S = 1, Co = 0$, 符合预期。

当 $A = 1, B = 0, C = 0, Ctrl = 1$ 进行减法运算时, 此时代表存在借位, 此位足够借位, $S = 0, Co = 1$, 符合预期。

4.1.2 四位全加器

1. 新建源文件, 类型是 Schametic, 文件名称用 AddSub4b, 原理图方式进行设计
2. 进行波形仿真
3. 创建 AddSub4b 的 symbol

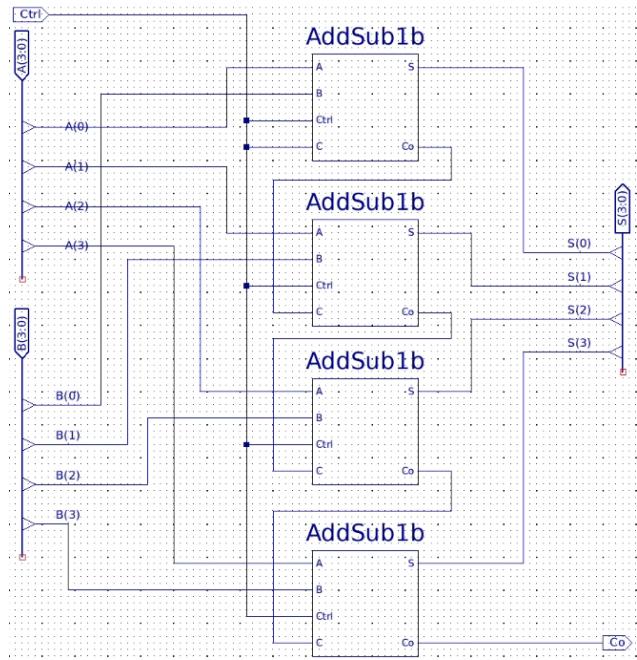


图 12: AddSub4b 原理图

仿真激励代码:

```

1  initial begin
2      A = 0;
3      B = 0;
4      Ctrl = 0;
5
6      A = 4'b1011;
7      B = 4'b0010;
8      #50;
9      B = 4'b0001;
10     #50;
11     A = 4'b1010;
12     B = 4'b0011;
13     Ctrl = 1;
14     #50
15     B = 4'b0100;
16     #50
17     A = 0;
18     B = 0;
19     Ctrl = 0;
20 end

```

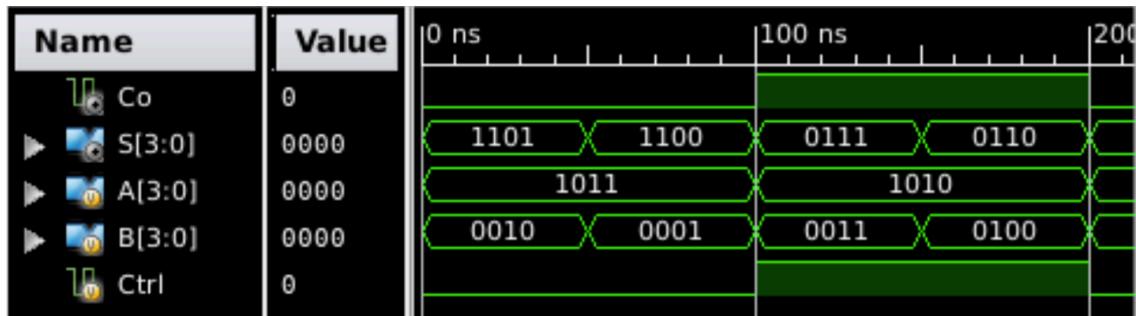


图 13: AddSub4b 波形仿真

其中第一组测试中 $A = 1011_2 = 11_{10}$, $B = 0010_2 = 2_{10}$, $Ctrl$ 为 0 进行加法运算, $S = A + B = 1011_2 + 0010_2 = 1101_2 = 13_{10}$, 符合预期。

第三组测试中 $A = 1010_2 = 10_{10}$, $B = 0011_2 = 3_{10}$, $Ctrl$ 为 1 进行减法运算, $S = A - B = 1010_2 - 0011_2 = 0111_2 = 7_{10}$, 符合预期。

创建 symbol 如下：

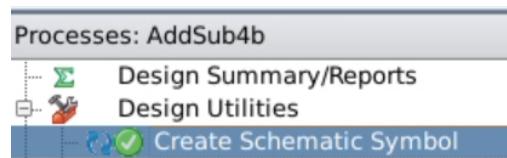


图 14: AddSub4b symbol

4.2 任务 2：实现 4 位 ALU 及应用设计

1. 新建工程 MyALU, Top Level Source Type 选择 HDL
2. 新建源文件, 类型是 Verilog, 文件名 Top, 右键设为“Set as Top Module”

4.2.1 ALU 电路设计

1. 新建源文件, 类型是 Schametic, 文件名称是 myALU, 原理图方式进行设计
2. 进行波形仿真

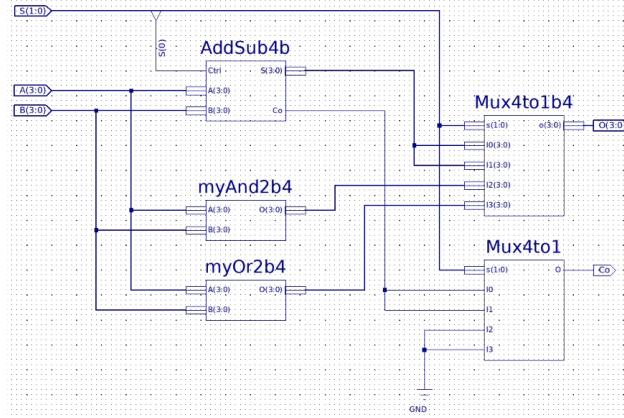


图 15: ALU 原理图

仿真激励代码:

```

1  initial begin
2      A = 0;
3      B = 0;
4      S = 0;
5
6      A = 4'b1010;
7      B = 4'b0111;
8      #50;
9      B = 4'b0011;
10     #50;
11     S = 2'b01;
12     #50
13     S = 2'b10;
14     #50;
15     S = 2'b11;
16     #50
17     A = 0;
18     B = 0;
19     S = 0;
20
end

```

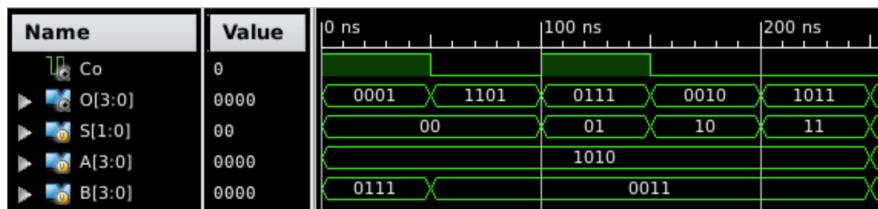


图 16: ALU 波形仿真

其中第一组测试中 $A = 1010_2 = 10_{10}$, $B = 0111_2 = 7_{10}$, S 为 00 进行加法运算, $A + B =$

$1010_2 + 0111_2 = 10001_2 = 17_{10}$, $S = 0001_2 = 1_{10}$, $Co = 1$, 符合预期。

第二组测试中 $A = 1010_2 = 10_{10}$, $B = 0011_2 = 3_{10}$, S 为 00 进行加法运算, $S = A + B = 1010_2 + 0011_2 = 1101_2 = 13_{10}$, $Co = 0$, 符合预期。

第三组测试中 $A = 1010_2 = 10_{10}$, $B = 0011_2 = 3_{10}$, S 为 01 进行减法运算, $S = A - B = 1010_2 - 0011_2 = 0111_2 = 7_{10}$, 符合预期。

第四组测试中 $A = 1010_2 = 10_{10}$, $B = 0011_2 = 3_{10}$, S 为 10 进行与运算, $S = A \& B = 0010_2 = 2_{10}$, 符合预期。

第五组测试中 $A = 1010_2 = 10_{10}$, $B = 0011_2 = 3_{10}$, S 为 11 进行或运算, $S = A | B = 1011_2 = 11_{10}$, 符合预期。

4.2.2 顶层模块 Top 设计

在 Top.v 中用行为描述进行设计

1. 实例化 pbdebounce 模块对 2 个按键进行去抖
2. 实例化 AddSub4b 模块实现 4 位加减法
3. 实例化 clkdiv 模块, 提供 1ms 时钟
4. 用 num[3:0] 表示 A, 用 num[7:4] 表示 B
5. 实例化 CreateNumber 模块, 用 2 个按键对 num[7:4]、num[3:0] 自增或自减
6. 实例化 DispNum 模块, 显示 A、B、Co、C

```
1 `timescale 1ns / 1ps
2 module top(
3     input wire clk,
4     input wire [1:0] BTN,
5     input wire [1:0] SW1,
6     input wire [1:0] SW2,
7     output wire [3:0] AN,
8     output wire [7:0] SEGMENT,
9     output wire BTNX4
10 );
11
12     wire [15:0] num;
13     wire [1:0] btn_out;
14     wire [3:0] C;
15     wire Co;
16     wire [31:0] clk_div;
17     wire [15:0] disp_hexs;
18
19     assign disp_hexs[15:12] = num[3:0];
20     assign disp_hexs[11:8] = num[7:4];
```

```

21      assign disp_hexs[7:4] = {3'b000, Co};
22      assign disp_hexs[3:0] = C[3:0];
23
24      pbdebounce m0(clk_div[17], BTN[0], btn_out[0]);
25      pbdebounce m1(clk_div[17], BTN[1], btn_out[1]);
26      clkdiv m2(.clk(clk),
27                  .rst(1'b0),
28                  .clkdiv(clk_div));
29      CreateNumber m3(btn_out, SW1, num);
30      myALU m5(.S(SW2),
31                  .A(num[3:0]),
32                  .B(num[7:4]),
33                  .O(C[3:0]),
34                  .Co(Co));
35      DispNum m6(.clk(clk),
36                  .HEXS(disp_hexs),
37                  .LES(4'b0),
38                  .point(4'b0),
39                  .RST(1'b0),
40                  .AN(AN),
41                  .Segment(SEGMENT));
42
43      assign BTNX4 = 1'b0;
44
45 endmodule

```

其中 pbdebounce 模块使用实验原理一节中的代码实现，clkdiv 模块在之前的实验中已经实现。

4.2.3 物理验证

UCF 引脚定义：

```

1  NET "clk"    LOC = AC18 | IOSTANDARD = LVCMOS18;
2  NET "SW1[1]" LOC = AA10 | IOSTANDARD = LVCMOS15;
3  NET "SW1[0]" LOC = AB10 | IOSTANDARD = LVCMOS15;
4  NET "SW2[0]" LOC = AF13 | IOSTANDARD = LVCMOS15;
5  NET "SW2[1]" LOC = AF10 | IOSTANDARD = LVCMOS15;
6  NET "SEGMENT[0]" LOC = AB22 | IOSTANDARD = LVCMOS33;
7  NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;
8  NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;
9  NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;
10 NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;
11 NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;
12 NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;
13 NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;
14 NET "AN[3]"   LOC = AC22 | IOSTANDARD = LVCMOS33;

```

```
15 NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;  
16 NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;  
17 NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;  
18 NET "btn[0]" LOC = V14 | IOSTANDARD = LVCMOS18;  
19 NET "btn[1]" LOC = W14 | IOSTANDARD = LVCMOS18;  
20 NET "BTNX4" LOC = W16 | IOSTANDARD = LVCMOS18;
```

Generate Programming File，下载到开发板上，按照实验要求进行测试。

测试按键加减方向控制 SW1[1] 控制 A，SW1[0] 控制 B。开关断开时，每按一次按钮，对应数字应当加一；开关闭合时，每按一次按钮，对应数字应当减一。

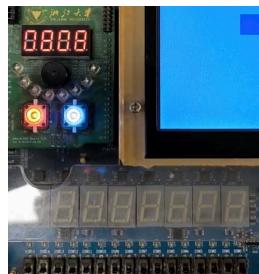


(a) 按下 BTN1 前



(b) 按下 BTN1 后，A 增加 1

图 17: SW1[1] 断开



(a) 按下 BTN1 前



(b) 按下 BTN1 后，A 减小 1

图 18: SW1[1] 闭合

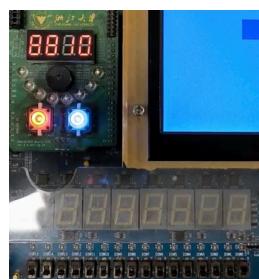


(a) 按下 BTN2 前

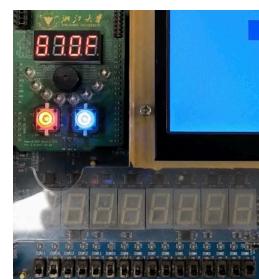


(b) 按下 BTN2 后, B 增加 1

图 19: SW1[0] 断开



(a) 按下 BTN2 前



(b) 按下 BTN2 后, B 减小 1

图 20: SW1[0] 闭合

测试 ALU 运算控制



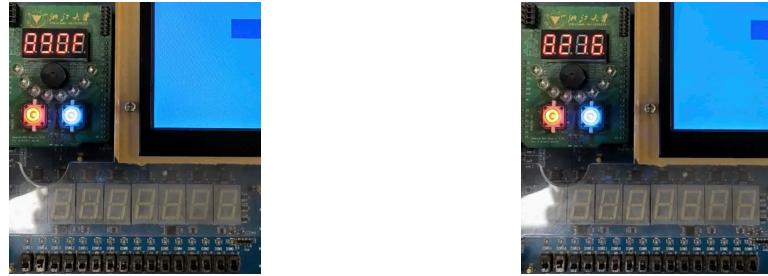
(a) SW = 00, A = 8, B = 7, C = 0, S = F (b) SW = 00, A = F, B = 8, C = 1, S = 7



图 21: 加法

此时为加法, 图 (a) 中 $A = 8, B = 7, A + B = 15 = F$, 显示 $C = 0, S = F$, 符合预期。

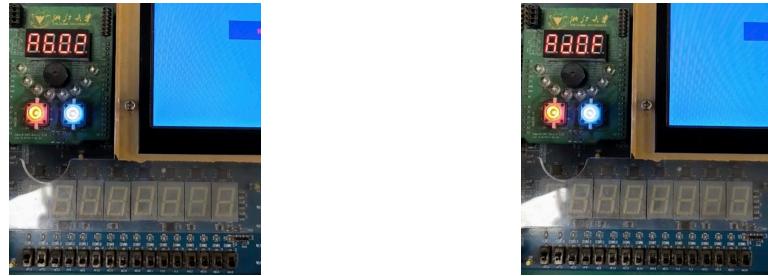
图 (b) 中 $A = F, B = 8, A + B = 23$, 显示 $C = 1, S = 7$, 符合预期。



(a) $SW = 01, A = 8, B = 9, C = 0, S = F$ (b) $SW = 01, A = 8, B = 2, C = 1, S = 6$

图 22: 减法

此时为减法, 图 (a) 中 $A = 8, B = 9, A - B = -1 = F$, 显示 $C = 0, S = F$, 符合预期。图 (b) 中 $A = 8, B = 2, A - B = 6$, 显示 $C = 1, S = 6$, 符合预期。



(a) $SW = 10, A = A, B = 6, C = 0, S = 2$ (b) $SW = 11, A = A, B = D, C = 0, S = F$

图 23: 与、或操作

图 (a) 为与操作, $A = A \& B = 10 \& 6 = 2$, 显示 $C = 0, S = 2$, 符合预期。

图 (b) 为或操作, $A = A | B = 10 | 13 = 15 = F$, 显示 $C = 0, S = F$, 符合预期。

5 实验结果分析

仿真波形和上板测试的结果在上文都进行了分析, 均符合预期。

6 讨论与心得

本次实验实现了 4 位加减法器和 4 位 ALU, 巩固了 Verilog 语言的使用, 对数字电路的实现有了更深的理解。

其中防抖动模块解决了上次实验中按钮按下后判定多次的问题, 使得按键输入更加稳定。