浙江大学

本科实验报告

课程名称：计算机逻辑设计基础

姓　　名：刘晨

学　　院：计算机科学与技术学院

专　　业：图灵班

学　　号：3190104666

指导教师：董亚波

2020 年 11 月 9 日

浙江大学实验报告

课程名称： 计算机逻辑设计基础 实验类型： 综合

实验项目名称： 加法器、加减法器和 ALU 基本原理与设计

学生姓名： 刘晨 专业： 图灵 1901 学号： 3190104666

同组学生姓名： 指导老师： 董亚波

实验地点： 东 4-509 实验日期：2020 年 11 月 9 日

一、实验目的和要求

掌握一位全加器的工作原理和逻辑功能

掌握串行进位加法器的工作原理和进位延迟

掌握减法器的实现原理

掌握加减法器的设计方法

掌握 ALU 基本原理及在 CPU 中的作用

掌握 ALU 的设计方法

二、实验内容和原理

内容：

任务 1：原理图方式设计 4 位加减法器

任务 2：实现 4 位 ALU 及应用设计

原理：

任务 1：

1 位全加器原理：

三个输入位：数据位 Ai 和 Bi，低位进位输入 Ci

二个输出位：全加和 Si，进位输出 Ci+1

多位串行进位加法器原理：

由一位全加器将进位串接构成

低位进位 C0 为 0，Ci 为高位进位输出

1 位加减法器原理：

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 C0 为 1

多位串行进位全减器原理：

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 C0 为 1

任务 2：

4 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 & B

S(1:0) = 11：C = A | B

ALU 计算得到进位 Co 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

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 10~20ms

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

三、实验过程和数据记录

任务 1：原理图方式设计 4 位加减法器

1. 一位全加器的设计

```
21  module adder_1bit(
22      input wire a, b, ci,
23      output wire s, co);
24      and m0(c1,a,b);
25      and m1(c2,b,ci);
26      and m2(c3,a,ci);
27      xor m3(s1,a,b);
28      xor m4(s,s1,ci);
29      or  m5(co,c1,c2,c3);
30  endmodule
```

图 1 一位全加器的设计代码

2. 一位加减法器的设计

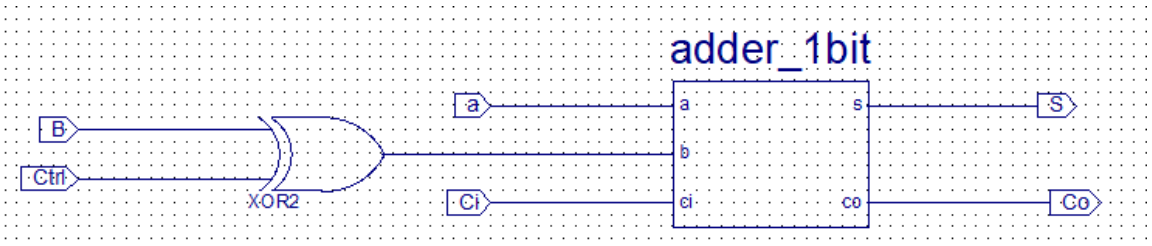

图 2 一位加减法器的设计原理图

3. 四位加法器的设计

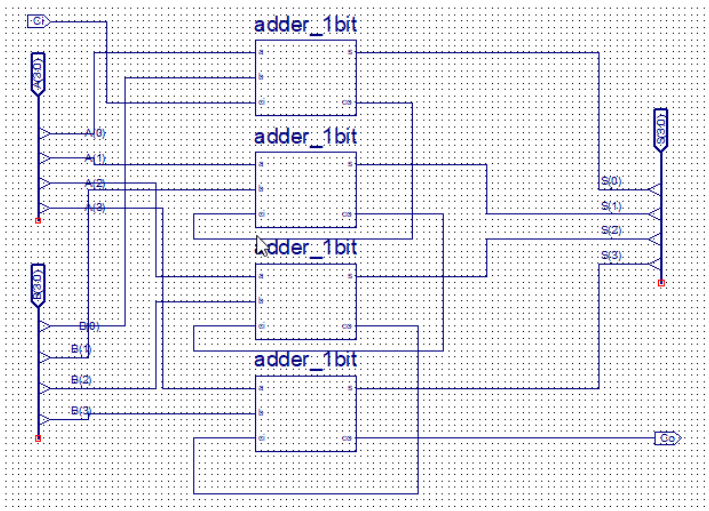

图 3 四位加法器的设计原理图

4. 四位加减法器的设计

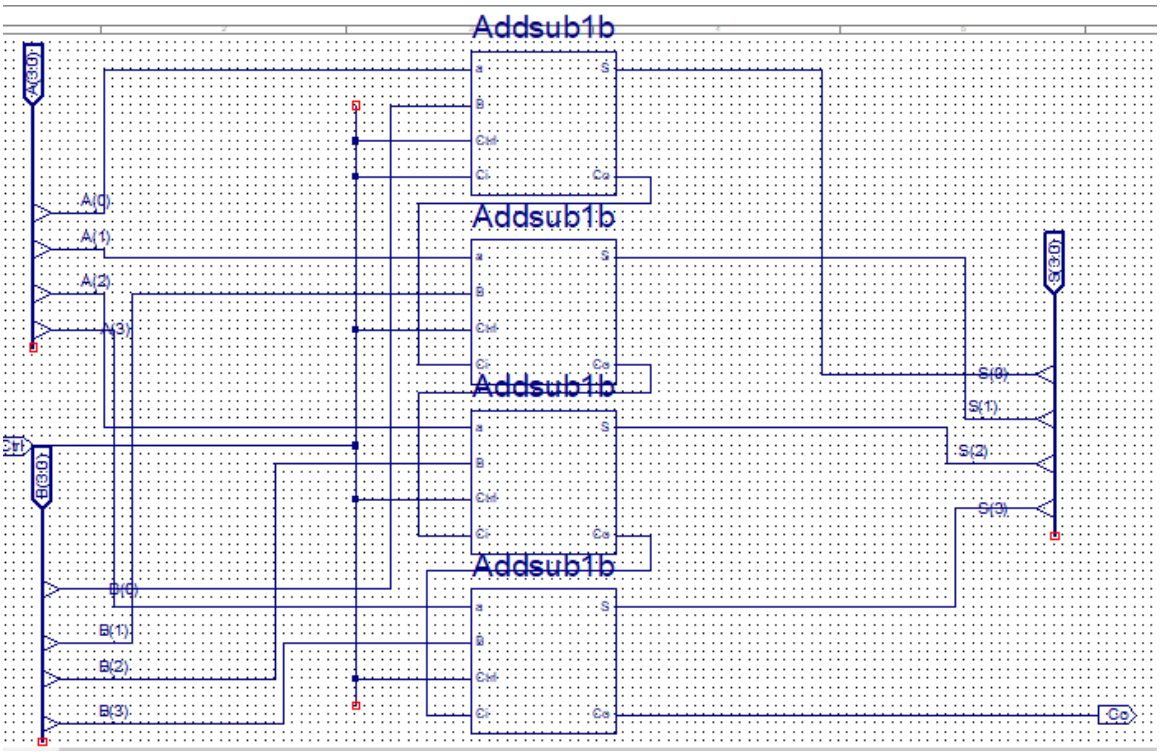

图 4 四位加减法器的设计原理图

任务 2：实现 4 位 ALU 及应用设计

1. 4 位 2 输入与门的设计

```
21  `timescale 1ns / 1ps
22
23  module myAnd2b4(A,
24                  B,
25                  C);
26
27      input [3:0] A;
28    input [3:0] B;
29    output [3:0] C;
30
31
32    AND2  XLXI_1 (.I0(B[0]),
33                  .I1(A[0]),
34                  .O(C[0]));
35    AND2  XLXI_2 (.I0(B[1]),
36                  .I1(A[1]),
37                  .O(C[1]));
38    AND2  XLXI_3 (.I0(B[2]),
39                  .I1(A[2]),
40                  .O(C[2]));
41    AND2  XLXI_4 (.I0(B[3]),
42                  .I1(A[3]),
43                  .O(C[3]));
44  endmodule
```

图 5 4 位 2 输入与门的设计的 verliog 代码

2.4 位 2 输入或门的设计

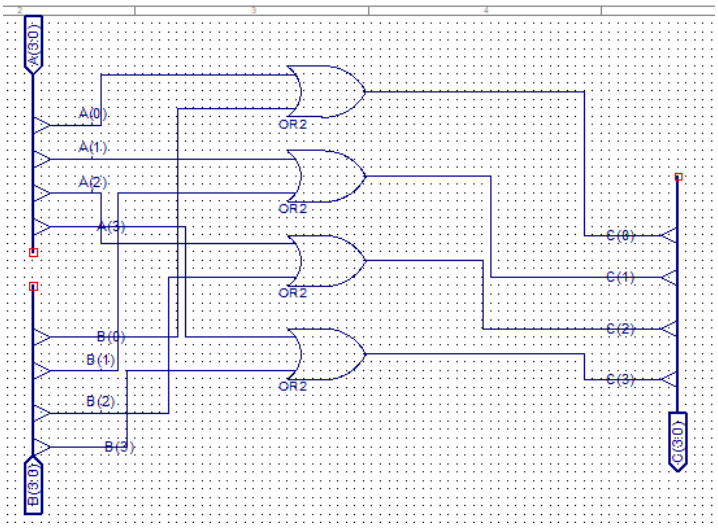

图 6　4 位 2 输入或门的逻辑原理图

3.MyALU 设计及仿真

设计：

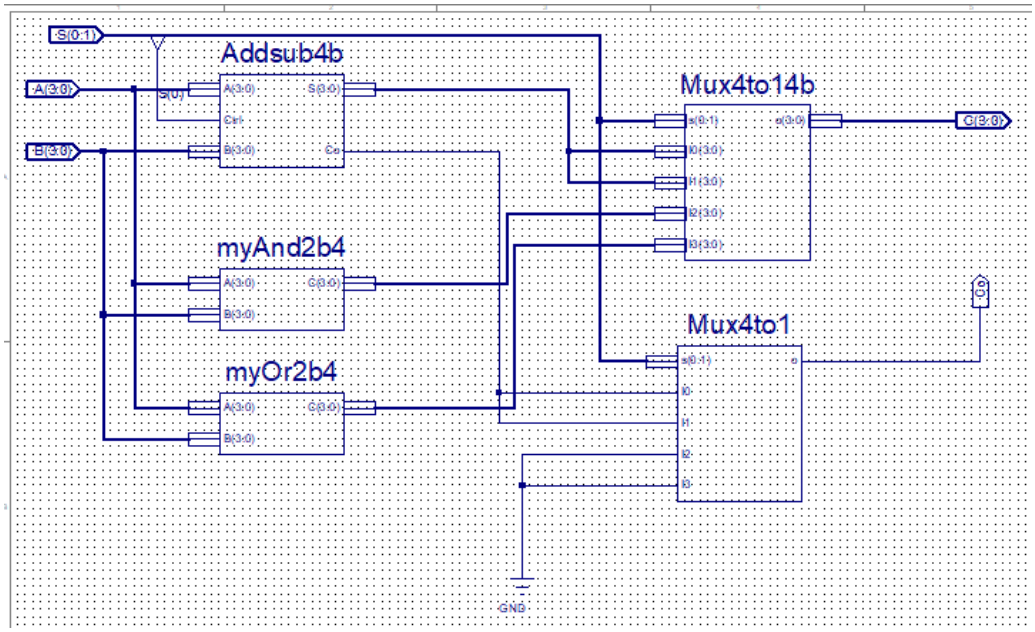

图 7　myALU 设计原理图

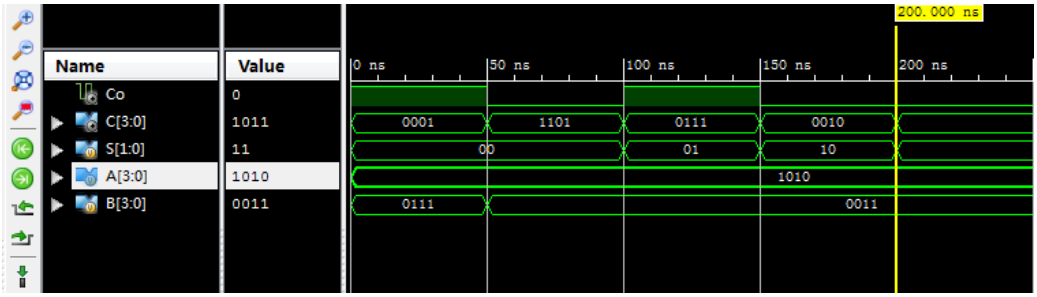

图 8　myALU 仿真结果

4. 设计按键数据输入模块

```
21  module CreateNumber(
22      input wire [3:0] btn,
23      input wire [3:0] sw,
24      output reg [15:0] num
25      );
26      wire [3:0] A,B,C,D;
27
28      initial num <= 16'b1010_1011_1100_1101;
29
30      Addsub4b a1(.A(num[3:0]),.B(4'b0001),.Ctrl(sw[0]),.S(A));
31      Addsub4b a2(.A(num[3:0]),.B(4'b0001),.Ctrl(sw[0]),.S(B));
32
33      //assign C = num[11:8] + 4'd1;
34      //assign D = num[15:12] + 4'd1;
35
36      always@(posedge btn[0]) num[3:0] <= A;
37      always@(posedge btn[1]) num[7:4] <= B;
38      //always@(posedge btn[2]) num[11:8] <= C;
39      //always@(posedge btn[3]) num[15:12] <= D;
40
41  endmodule
```

图 9　CreateNumber 模块

5. 设置按键去抖动

```
17  module pbdebounce(
18     input wire clk_1ms,
19     input wire button,
20     output reg pbreg
21     );
22
23     reg [7:0] pbshift;
24
25     always@(posedge clk_1ms) begin
26        pbshift=pbshift<<1;
27        pbshift[0]=button;
28        if (pbshift==8'b0)
29           pbreg=0;
30        if (pbshift==8'hFF)
31           pbreg=1;
32     end
33  endmodule
```

图 10　pbdebounce 模块设计代码

6. 设置时钟模块

```
21  module clkdiv(
22     input clk,
23     input rst,
24     output reg[31:0]clkdiv
25     );
26    always @ (posedge clk or posedge rst) begin
27       if (rst) clkdiv <= 0;
28       else clkdiv <= clkdiv + 1'b1;
29    end
30  endmodule
```

图 11 时钟模块设计代码

7.DispNum 模块的设计

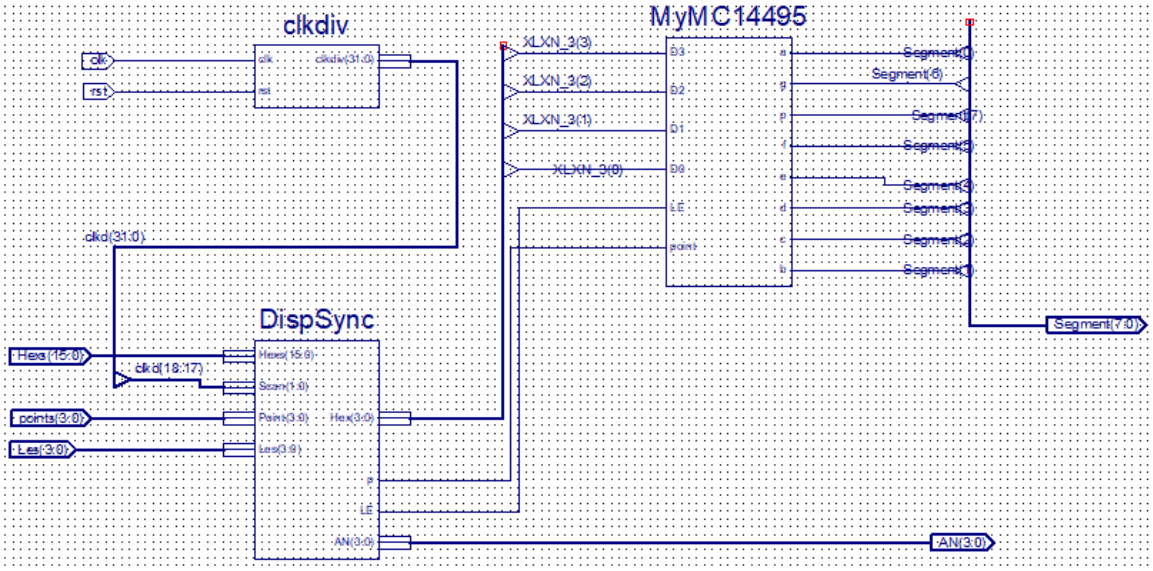

图 13 DispNum 设计原理图

8.Top 顶层模块的设计

```
module top(
   input wire clk,
   input wire [1:0]BTN,
   input wire [1:0]SW1,
   input wire [1:0]SW2,
   output wire [3:0]AN,
   output wire [7:0]SEGMENT,
   output wire BTNX4
   );
   wire [15:0] num;
   wire [1:0] btn_out;
   wire [3:0] C;
   wire Co;
   wire [31:0] clk_div;
   wire [15:0] disp_hexs;
   assign disp_hexs[15:12] = num[3:0]; //A
   assign disp_hexs[11:8] = num[7:4];  //B
   assign disp_hexs[7:4] = {3'b000, Co};
   assign disp_hexs[3:0] = C[3:0];

   pbdebounce m0(clk_div[17], BTN [0], btn_out[0]);
   pbdebounce m1(clk_div[17], BTN[1], btn_out[1]);
   clkdiv m2(.clk(clk),.rst(1'b0),.clkdiv(clk_div));
   CreateNumber m3(.btn({1'b0,1'b0,btn_out[0],btn_out[1]}),.sw({1'b0,1'b0,SW1[0],SW1[1]}),.num(num));
   myALU m5(.A(num[3:0]),.B(num[7:4]),.S(SW2),.C(C),.Co(Co));
   dispnum m6(.clk(clk), .Hexs(disp_hexs), .Les(4'b0), .points(4'b0),.rst(1'b0), .AN(AN), .Segment(SEGMENT));
   assign BTNX4 = 1'b0; //Enable button inputs
endmodule
```

图 14 Top 顶层模块的设计代码

9.引入 ucf 文件，在实验板上面模拟

```
NET "btn[0]" LOC = W14 | IOSTANDARD = LVCMOS18;
NET "btn[0]" CLOCK_DEDICATED_ROUTE = FALSE;
NET "btn[1]" LOC = V14 | IOSTANDARD = LVCMOS18;
NET "btn[1]" CLOCK_DEDICATED_ROUTE = FALSE;

NET "BTNX4" LOC = W16 | IOSTANDARD = LVCMOS18;

NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;

NET "SW1[1]"LOC = AA10 | IOSTANDARD = LVCMOS15;
NET "SW1[0]"LOC = AB10 | IOSTANDARD = LVCMOS15;
NET "SW2[1]"LOC = AA13 | IOSTANDARD = LVCMOS15;
NET "SW2[0]"LOC = AA12 | IOSTANDARD = LVCMOS15;

NET "SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;

NET "AN[0]" LOC = AD21 | IOSTANDARD =LVCMOS33;
NET "AN[1]" LOC = AC21 | IOSTANDARD =LVCMOS33;
NET "AN[2]" LOC = AB21 | IOSTANDARD =LVCMOS33;
NET "AN[3]" LOC = AC22 | IOSTANDARD =LVCMOS33;
```

图 15　ucf 文件代码

实验结果如下：

如果 S0=0，那么按按钮 1，数字板上的第一个数字增大 1；

如果 S0=1，那么按按钮 1，数字板上的第一个数字减小 1；

如果 S1=0，那么按按钮 2，数字板上的第二个数字增大 1；

如果 S1=1，那么按按钮 2，数字板上的第二个数字减小 1；

S2，S3 负责控制进行的运算。

S2	S3	进行运算
0	0	+
0	1	-
1	0	&
1	1	|

调节前两个数字分别为 A，B，拨动开关 S2，S3，结果如下：

S2	S3	结果	原因
0	0	AB15	A+B=1010+1011=10101(15)
0	1	AB0F	1A-B=11010-1011=1111(F)
1	0	AB0A	A&B=1010&1011=1010(A)
1	1	AB0B	A|B=1010|1011=1011(B)

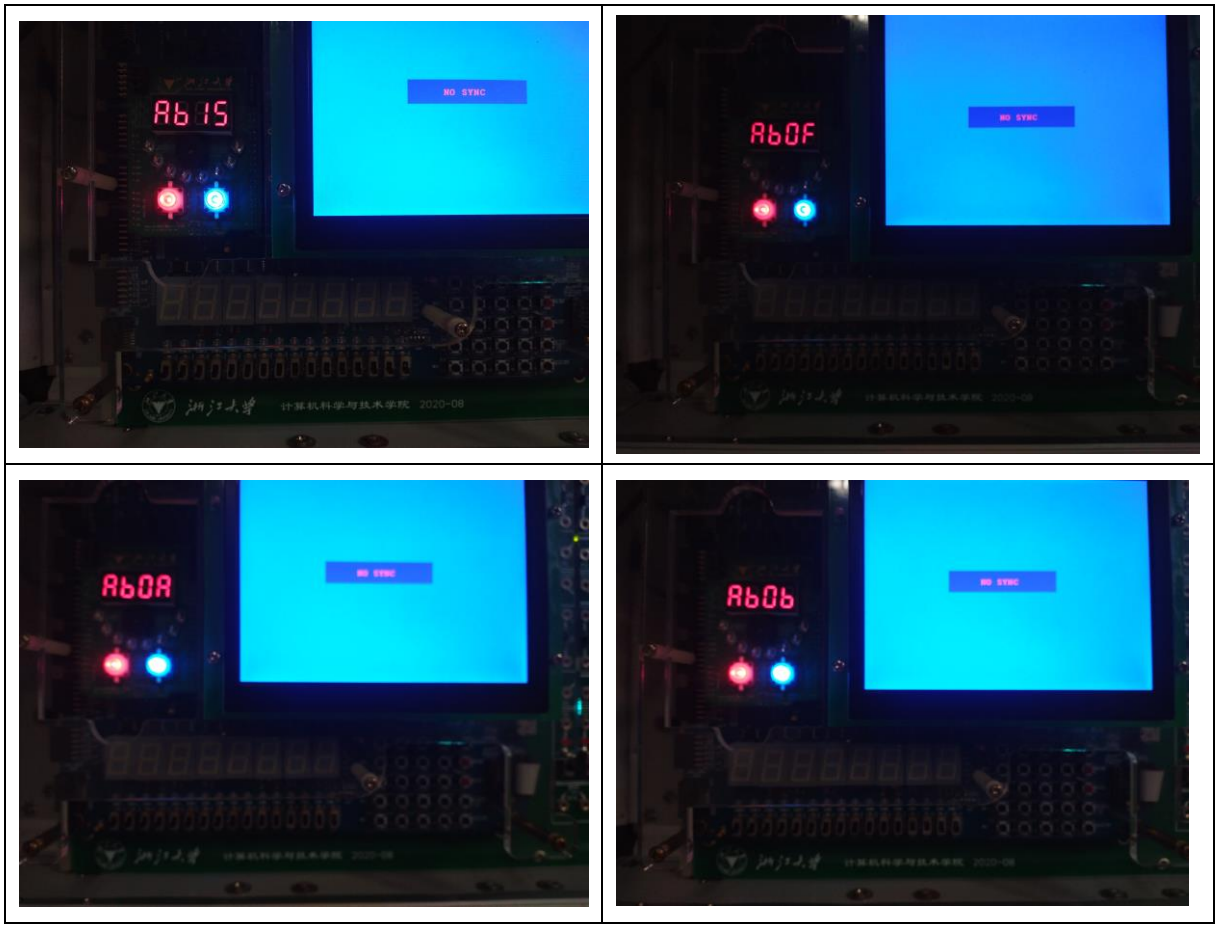

图 16 结果图（AB）

注：结果图从左到右，从上到下分别是开关位于 00,01,10,11 的结果，以下同。

调节前两个数字分别为 7，8，拨动开关 S2，S3，结果如下：

S2	S3	结果	原因
0	0	780F	7+8=0111+1000=1111(F)
0	1	780F	17-8=10111-1000=1111(F)
1	0	7800	7&8=0111&1000=0000(0)

1	1	780F	7|8=0111|1000=1111（F）

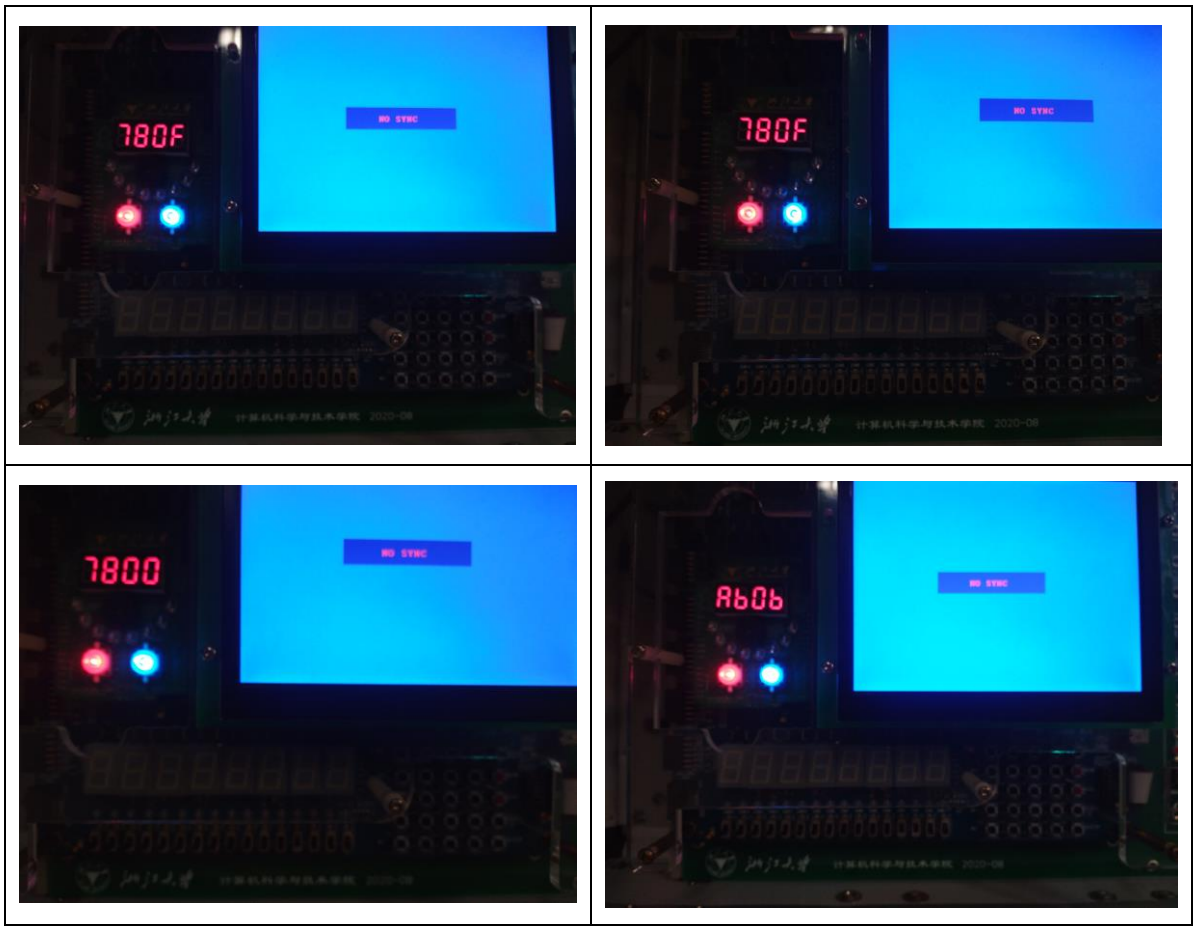

图 17 结果图（78）

调节前两个数字分别为 7，F，拨动开关 S2，S3，结果如下：

S2	S3	结果	原因
0	0	7F16	7+F=0111+1111=10110(16)
0	1	7F08	17-F=10111-1111=1000(8)
1	0	7F07	7&F=0111&1111=0111(7)
1	1	7F0F	7|F=0111|1111=1111（F）

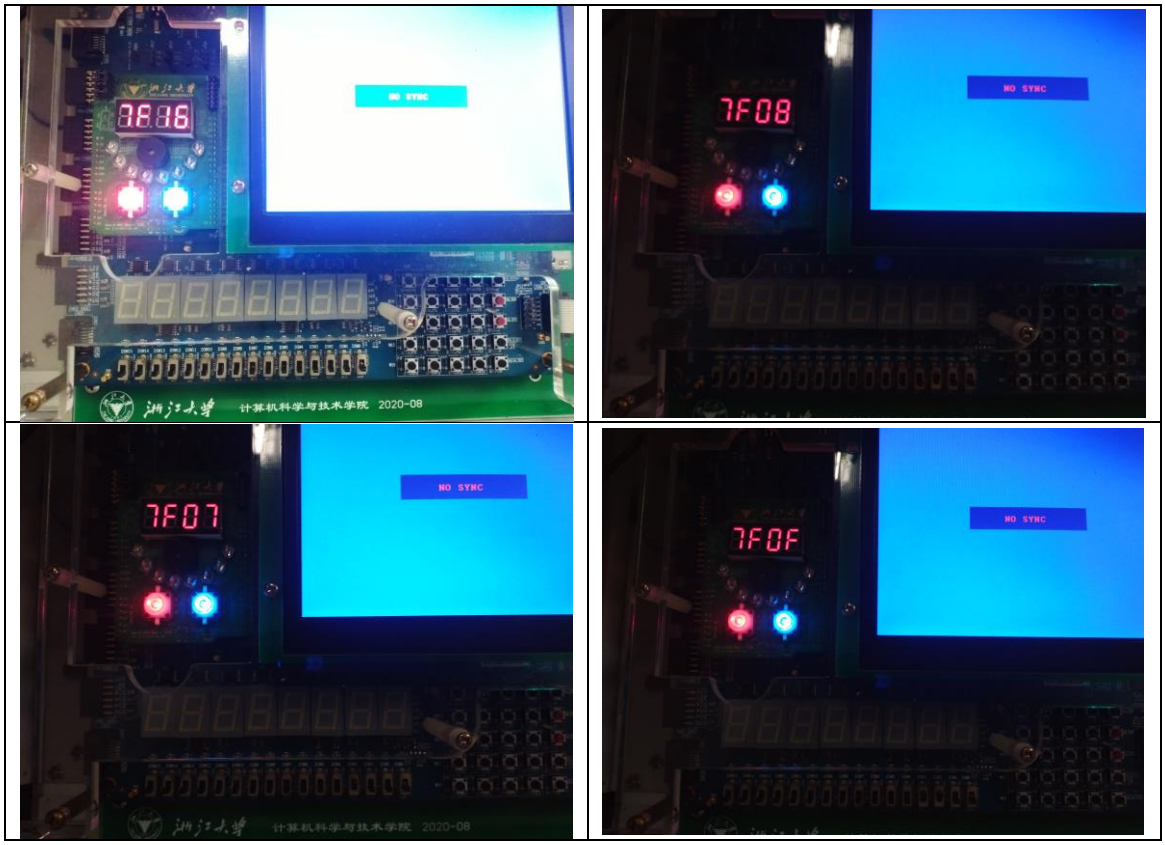

图 18 结果图（7F）

四、实验结果分析

总体和预期实验结果内容一致。

硬件描述代码和原理图所要实现的功能一致。

仿真结果符合预期。

Verliog 代码内容实现了预期功能。

实验板上面的结果实现了预期的运算功能。

五、讨论与心得

在画结构图的时候，出现了奇特的情况，之后原先应该是输出端的变成了输入端，并且查看硬件描述代码，出现了一个 S=dummy(3:0)。于是进行了思考，发现输出端和输入端在画线的时候并不确定，并不是原理图里面应该是输出的地方就会自动变成输出，如果发现出现了不正常的地方，应该点击来转换为输出。