

# 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:	王晓宇
学 院:	竺可桢学院
专 业:	计算机
邮 箱:	<a href="mailto:1657946908@qq.com">1657946908@qq.com</a>
QQ 号:	1657946908
电 话:	19550222634
指导教师:	洪奇军
报告日期:	2023 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： 王晓宇 学号： 3220104364 同组学生姓名：

实验地点： 紫金港东四 509 室 实验日期： 2023 年 11 月 9 日

## 一、操作方法与实验步骤

### A. 任务 1：原理图方式设计 4 位加减法器

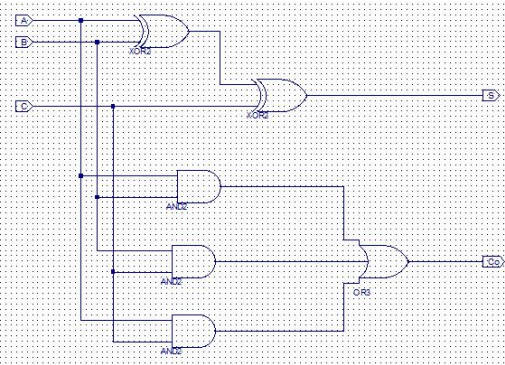
1 位全加器：

- 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  $C_{i+1}$

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

$$S_i = A_i \oplus B_i \oplus C_i$$
$$C_{i+1} = A_i B_i + B_i C_i + C_i A_i$$

根据一位全加器的输入输出关系，得到电路图

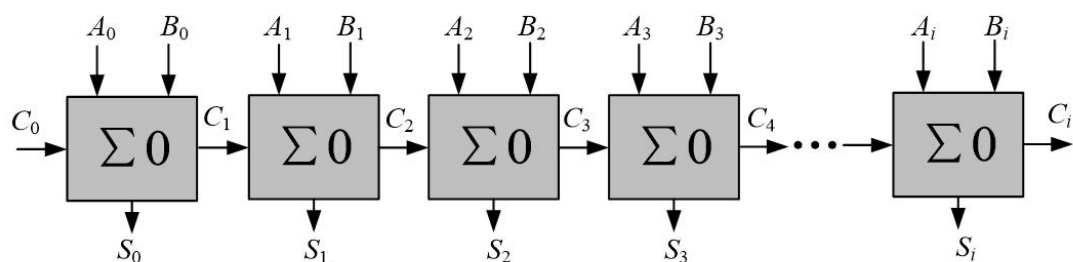


```
module adder_1bit(  
    input wire a, b, ci,  
    output wire s, co;  
    and m0(c1,a,b);  
    and m1(c2,b,ci);  
    and m2(c3,a,ci);  
    xor m3(s1,a,b);  
    xor m4(s,s1,ci);  
    or m5(co,c1,c2,c3);  
endmodule
```

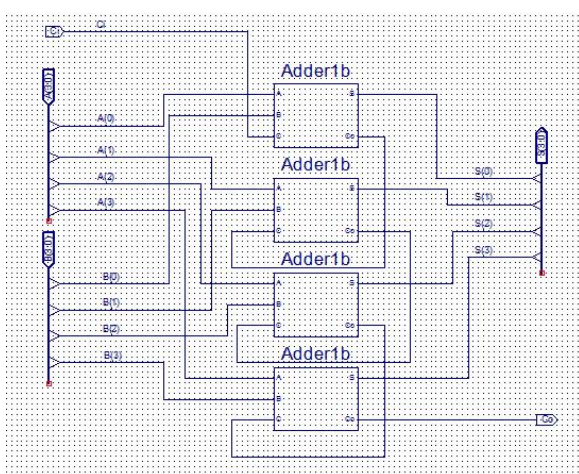
### 多位串行进位加法器

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

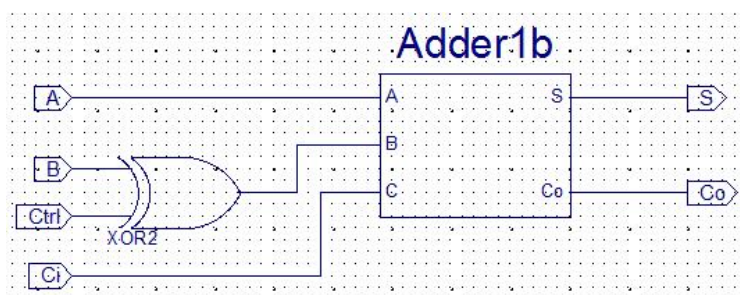


### 四位全加器的生成

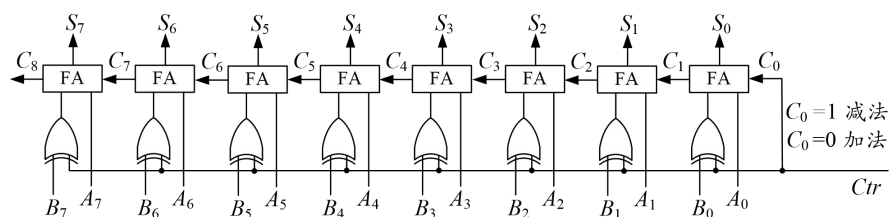


### 1 位加减法器

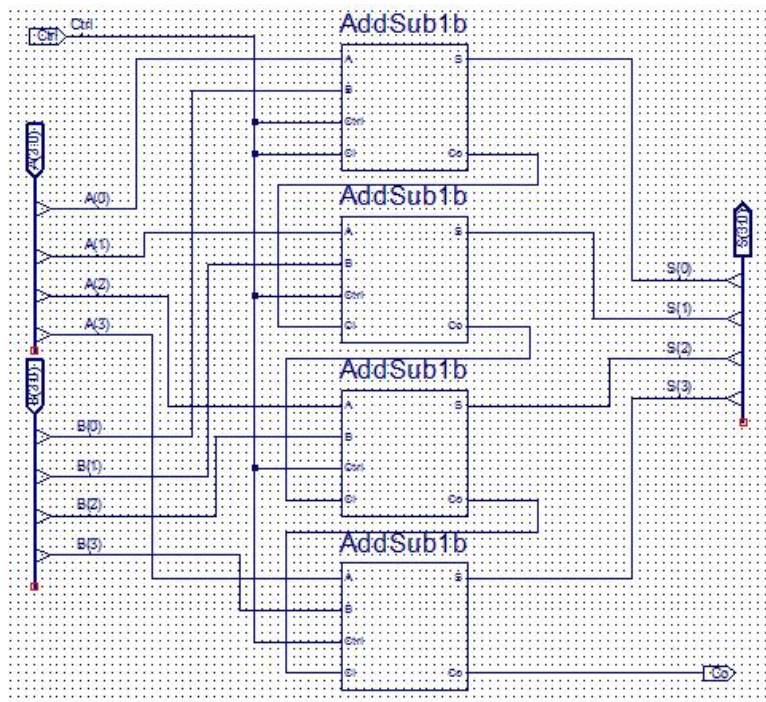
- 用负数补码加法实现，减数当作负数求补码
- 共用加法器
- 用“异或”门控制求反，低位进位  $C_0$  为 1



### 多位串行进位全减器



#### 4 位加减法器



#### 设计按键数据输入模块

```

module CreateNumber(
    input wire [3:0] btn,
    input wire [3:0] sw,
    output reg [15:0] num
);
    wire [3:0] A1,B1,C1,D1;

    initial num <= 16'b1010_1011_1100_1101; // display "AbCd"

    AddSub4b a1(.A(num[ 3: 0]),.B(4'b0001),.Ctrl(sw[0]),.S(A1));
    AddSub4b a2(.A(num[ 7: 4]),.B(4'b0001),.Ctrl(sw[1]),.S(B1));
    AddSub4b a3(.A(num[11: 8]),.B(4'b0001),.Ctrl(sw[2]),.S(C1));
    AddSub4b a4(.A(num[15:12]),.B(4'b0001),.Ctrl(sw[3]),.S(D1));

    always@(posedge btn[0]) num[ 3: 0] <= A1;
    always@(posedge btn[1]) num[ 7: 4] <= B1;
    always@(posedge btn[2]) num[11: 8] <= C1;
    always@(posedge btn[3]) num[15:12] <= D1;

endmodule

```

但其中的 C/D 可以不用赋值，在本节中，只需改变 AB 的值即可。

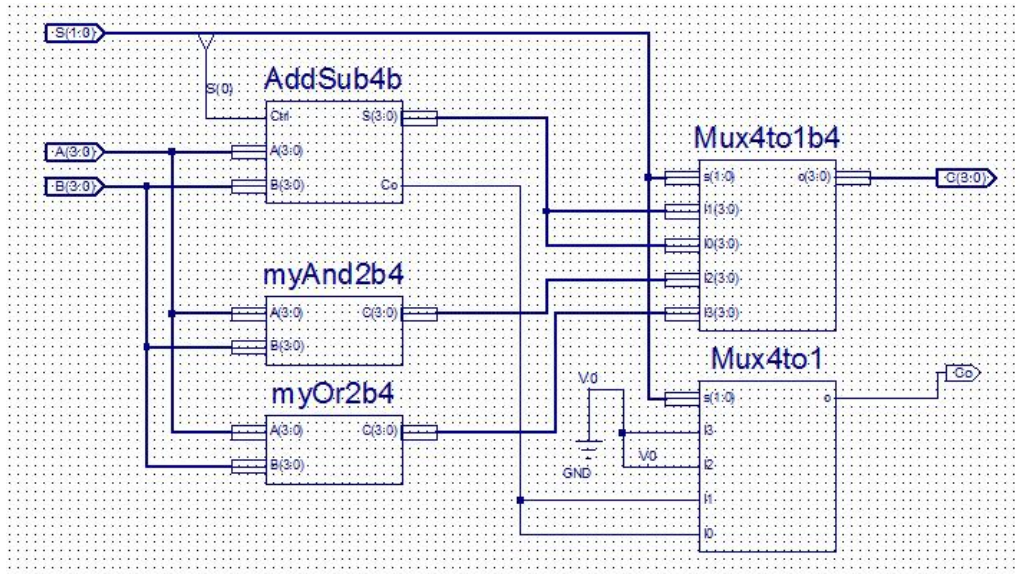
#### 防抖动模块 + 分频器

```

//
// Create Date: 18:27:10 11/08/2023
// Design Name:
// Module Name: pbdebounce
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
// =====
module pbdebounce(
    input wire clk_1ms,
    input wire button,
    output reg pdreg
);
    reg [7:0] pdshift;
    always @(posedge clk_1ms) begin
        pdshift <= pdshift < 1;
        pdshift[0] = button;
        if (pdshift == 8'b0)
            pdreg = 0;
        if (pdshift == 8'hFF)
            pdreg = 1;
    end
endmodule

```

## B. 任务 2：实现 4 位 ALU 及应用设计



```
// Bidirs
// Instantiate the UUT
ALU UUT (
    .A(A),
    .B(B),
    .S(S),
    .C(C),
    .Co(Co)
);
// Initialize Inputs
//`ifdef auto_init
integer i,j;
initial begin
    A = 4'b0000;
    B = 4'b1010;
    S = 0;
    for(i=0;i<=4;i=i+1)begin
        A=A+i;
        for (j = 0;j<=3;j=j+1)begin
            S = j;
            #100;
        end
    end
end
// `endif
end
endmodule
```

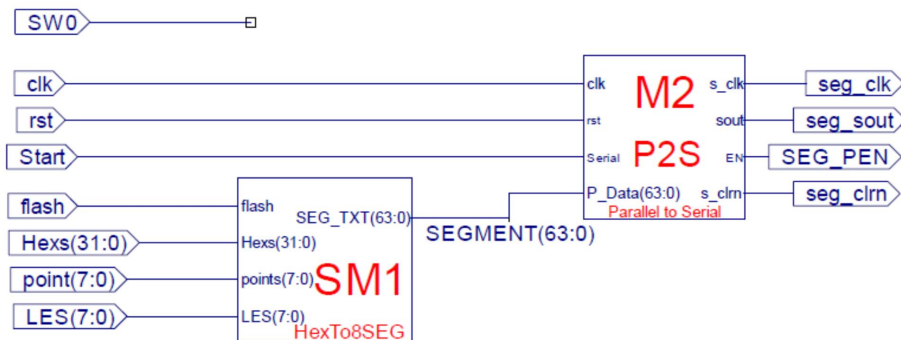
设计 8 位七段码显示模块

1. 用 Arduino Sword-002 子板四位动态扫描扩展
2. 用主板调用 P2S 模块输出静态显示

顶层模块名: Hex827Seg\_sch

1. 原理图输入
2. 调用模块实现
  - 调用 MC14495(Hex27Seg 工程复制)
  - 调用辅助时钟分频模块, 符号: clkdiv.sym(制作)
  - 修改 4 位七段扫描同步输出模块, 符号: dispync32.sym
  - 或设计八位七段静态译码模块(HexTo8SEG8), 调用 P2S 输出





```

module HexTo8SEG(input [31:0] Hexs,          //端口变量说明与定义合并
//
input [2:0] Scan,
input [7:0] points,
input [7:0] LES,
input flash,
output[63:0] SEG_TXT
);

Hex2Seg HTS0(Hexs[31:28],LES[7],points[7],flash,SEG_TXT[7:0]);
Hex2Seg HTS1(Hexs[27:24],LES[6],points[6],flash,SEG_TXT[15:8]);
Hex2Seg HTS2(Hexs[23:20],LES[5],points[5],flash,SEG_TXT[23:16]);
Hex2Seg HTS3(Hexs[19:16],LES[4],points[4],flash,SEG_TXT[31:24]);

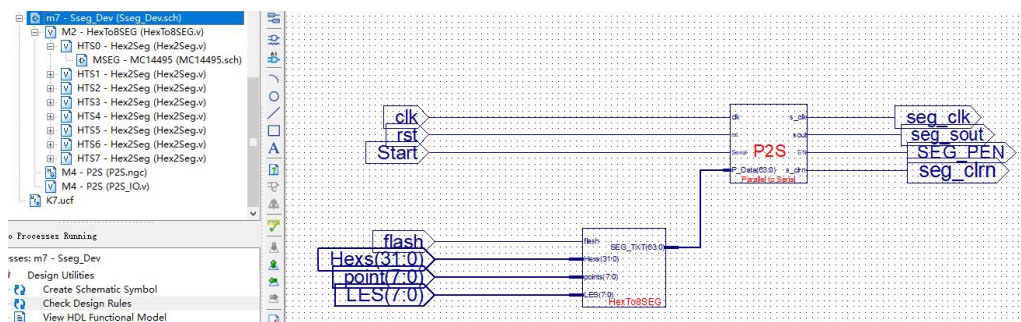
Hex2Seg HTS4(Hexs[15:12],LES[3],points[3],flash,SEG_TXT[39:32]);
Hex2Seg HTS5(Hexs[11:8], LES[2],points[2],flash,SEG_TXT[47:40]);
Hex2Seg HTS6(Hexs[7:4], LES[1],points[1],flash,SEG_TXT[55:48]);
Hex2Seg HTS7(Hexs[3:0], LES[0],points[0],flash,SEG_TXT[63:56]);

endmodule

module Hex2Seg(input [3:0] Hex,
input LE,
input point,
input flash,
output [7:0] Segment
);
wire en = LE & flash;
MC14495_ZJU MSEG(.D3(Hex[3]), .D2(Hex[2]), .D1(Hex[1]), .D0(Hex[0]), .LE(en), .POINT(point),
.a(a), .b(b), .c(c), .d(d), .e(e), .f(f), .g(g), .p(p));
assign Segment = {a, b, c, d, e, f, g, p};

endmodule

```



Top 实现:

```
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////////////////////////////////////////
module Top1
input wire clk,
input wire [1:0] BTN,
input wire [1:0] SW1,
input wire [1:0] SW2,
input wire [1:0] SW,
output wire [3:0] AN,
output wire [7:0] SEGMENT,
output wire BTNK4,
output wire seg_clk,
output wire seg_clrn,
output wire seg_sout,
output wire SEG_FEN
);

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];
assign disp_hexs[11:8] = num[7:4];
assign disp_hexs[7:4] = {3'b000,Co};
assign disp_hexs[3:0] = C[3:0];
assign BTNK4 = 1'b0;
clkdiv m2(clk,0,clk_div);
pbdebounce m0(clk_div[17],BTN[0],btn_out[0]);
pbdebounce m1(clk_div[17],BTN[1],btn_out[1]);
CreateNumber m3(btn_out,SW1,num);
ALU m5(.S(SW2[1:0]),.A(num[3:0]),.B(num[7:4]),.C(C),.Co(Co));
disp_num m6(.clk(clk),.HEXS(disp_hexs),.LES(4'b0000),.points(4'b0000),.RST(1'b1),.AN(AN),.segment(SEGMENT));
Sseg_Dev m7(.clk(clk),.rst(1'b0),.Start(clk_div[20]),.flash(clk_div[25]),.Hexs((disp_hexs,disp_hexs)),.point((4'b0000,SW[3:0])),.LES(SW[11

endmodule
```

引脚设置：

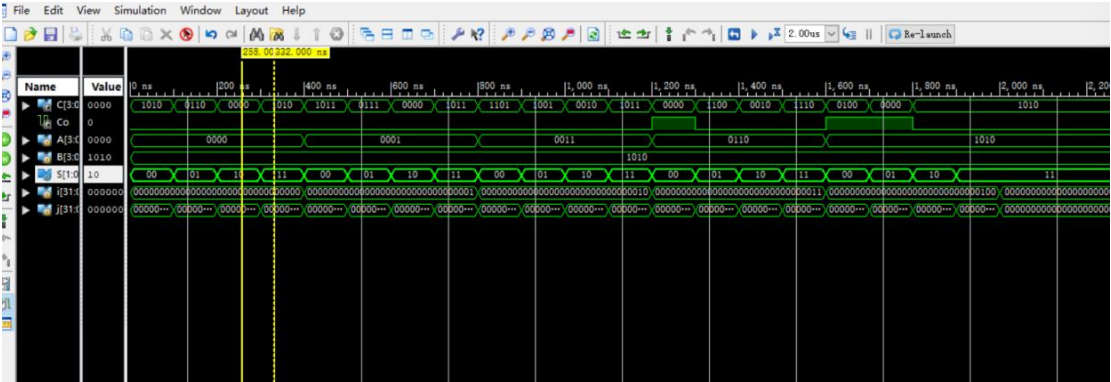
```
1 NET"clk"LOC = AC18 | IOSTANDARD=LVCMOS18;
2 NET"BTN[0]"LOC=W14 | IOSTANDARD=LVCMOS18;
3 NET"BTN[1]"clock_dedicated_route = false;
4 NET"BTN[1]"LOC=W14 | IOSTANDARD=LVCMOS18;
5 NET"BTN[1]"clock_dedicated_route = false;
6 NET"BTNK4"LOC=W16 | IOSTANDARD=LVCMOS18;
7
8 NET"SW1[0]"LOC=AF10 | IOSTANDARD=LVCMOS15;
9 NET"SW1[1]"LOC=AF13 | IOSTANDARD=LVCMOS15;
10 NET"SW2[0]"LOC=AE13 | IOSTANDARD=LVCMOS15;
11 NET"SW2[1]"LOC=AF8 | IOSTANDARD=LVCMOS15;
12
13 NET"SW[0]"LOC=AA10 | IOSTANDARD=LVCMOS15;
14 NET"SW[1]"LOC=AB10 | IOSTANDARD=LVCMOS15;
15 NET"SW[2]"LOC=AA13 | IOSTANDARD=LVCMOS15;
16 NET"SW[3]"LOC=AA12 | IOSTANDARD=LVCMOS15;
17 NET"SW[4]"LOC=Y13 | IOSTANDARD=LVCMOS15;
18 NET"SW[5]"LOC=Y12 | IOSTANDARD=LVCMOS15;
19 NET"SW[6]"LOC=AD11 | IOSTANDARD=LVCMOS15;
20 NET"SW[7]"LOC=AD10 | IOSTANDARD=LVCMOS15;
21 NET"SW[8]"LOC=AE10 | IOSTANDARD=LVCMOS15;
22 NET"SW[9]"LOC=AE12 | IOSTANDARD=LVCMOS15;
23 NET"SW[10]"LOC=AF12 | IOSTANDARD=LVCMOS15;
24 NET"SW[11]"LOC=AE8 | IOSTANDARD=LVCMOS15;
25
26 NET"SEGMENT[0]"LOC=AB22 | IOSTANDARD=LVCMOS33;#a
27 NET"SEGMENT[1]"LOC=AD24 | IOSTANDARD=LVCMOS33;#b
28 NET"SEGMENT[2]"LOC=AD23 | IOSTANDARD=LVCMOS33;#c
29 NET"SEGMENT[3]"LOC=Y21 | IOSTANDARD=LVCMOS33;#d
30 NET"SEGMENT[4]"LOC=W20 | IOSTANDARD=LVCMOS33;#e
31 NET"SEGMENT[5]"LOC=AC24 | IOSTANDARD=LVCMOS33;#f
32 NET"SEGMENT[6]"LOC=AC23 | IOSTANDARD=LVCMOS33;#g
33 NET"SEGMENT[7]"LOC=AA22 | IOSTANDARD=LVCMOS33;#point
34 NET"AN[0]"LOC=AD21 | IOSTANDARD=LVCMOS33;
35 NET"AN[1]"LOC=AC21 | IOSTANDARD=LVCMOS33;
36 NET"AN[2]"LOC=AB21 | IOSTANDARD=LVCMOS33;
37 NET"AN[3]"LOC=AC22 | IOSTANDARD=LVCMOS33;
38
39 NET"seg_clk" LOC = M24 | IOSTANDARD=LVCMOS33;
40 NET"seg_clrn" LOC = M20 | IOSTANDARD=LVCMOS33;
41 NET"seg_sout" LOC = L24 | IOSTANDARD=LVCMOS33;
42 NET"SEG_FEN" LOC = R18 | IOSTANDARD=LVCMOS33;
```

## 二、实验结果与分析

### 4 位 ALU 仿真

```
// Bidirs
// Instantiate the UUT
ALU UUT (
.A(A),
.B(B),
.S(S),
.C(C),
.Co(Co)
);
// Initialize Inputs
//`ifdef auto_init
integer i,j;
initial begin
A = 4'b0000;
B = 4'b1010;
S = 0;
for(i=0;i<4;i=i+1)begin
A=A+i;
for(j = 0;j<3;j=j+1)begin
S = j;
#100;
end
end
//`endif
end
endmodule
```


对于 A 的初值修改了 5 次分别为 4'b0000,4'b0001,4'b0011,4'b0110,4'b1010,每次进行加减与或运算，B 初值为 1010

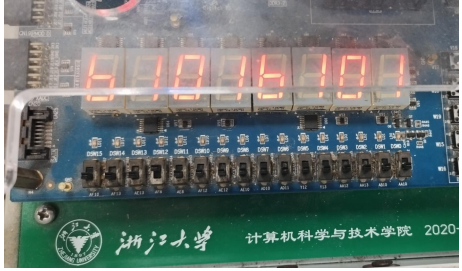
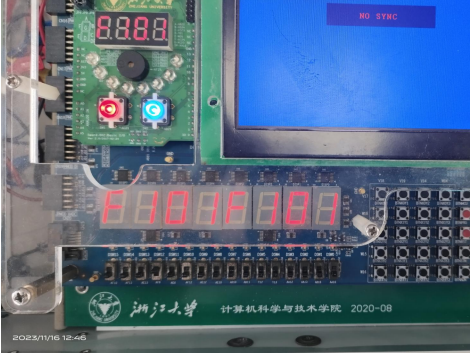


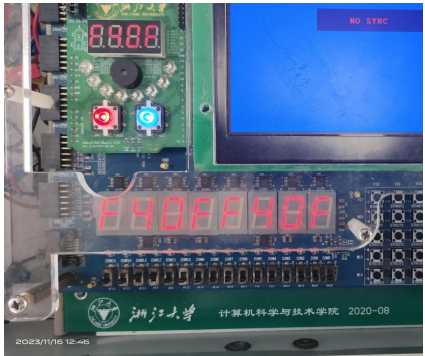
加法运算		
A/B	C0/C	实验图片
5/1	0/6	
8/8	1/0	


减法运算		
A/B	C0/C	实验图片
D/A	1/3	



D/5	1/8	
-----	-----	--

按位与运算		
A/B	C0/C	实验图片
B/1	0/1	
F/1	0/1	

按位或运算		
A/B	C0/C	实验图片
F/4	0/F	

F/B	O/F	
-----	-----	--

### 三、讨论、心得

这次的实验是要求先设计 ALU 基础逻辑单元，从底层开始设计，内含一位全加器 Adder（用 Verilog 语言编写）、四位全加器 Adder4b（使用 Adder 画图）、一位加减器 AddSub1b（使用 Adder 画图）、四位加减器 AddSub4b（使用 AddSub1b 画图）、防抖动模块 pbdebounce。在上节课的使用过的代码基础上，即使用 CreateNum、clk\_div、disp\_num 来使得 ABC0C 在七段数码管上刷新出现。

本节架构较为清楚,Top 模块负责统筹输入与输出，调用函数赋值。在本节实验中其他操作基本没有问题，只是在编写 Top 模块的时候报错，显示“对同一寄存器多次赋值”，这个确实应该是报 Error，但是最后发现该错误解决不了，只得重新 copy source 去新建 project 来查出哪里出了问题，结果重新编写了一份 Top 就可以正常编译了，有时 ISE 总会报一些离谱的错误，多尝试几次就能发现问题所在了，还有就是要学会去看他的报错提示。



VIVADO > SYNTHESIS



**andrew\_ruiz** (Member) asked a question.  
2011年12月18日 at 22:45

#### Signal is connected to multiple drivers...

I'm trying to synthesize my code but my error leads to the file quad\_dec.

The file in question is instantiated here, [lab5](#).

The error I get in synthesis is the following:

ERROR:Xst:528 - Multi-source in Unit <quad\_dec> on signal <nxt\_re\_state\_a<2>>; this signal is co

ERROR:Xst:528 - Multi-source in Unit <quad\_dec> on signal <nxt\_re\_state\_a<1>>; this signal is co

ERROR:Xst:528 - Multi-source in Unit <quad\_dec> on signal <nxt\_re\_state\_a<0>>; this signal is co