

EDA1 基于FPGA的简易计算器

李端 软件92 2019013268

EDA1 基于FPGA的简易计算器

- 一、实验目的
- 二、实验内容
 - 1.基本内容
 - 2.进一步研究内容
- 三、设计思路
 - 1、基本内容：十六进制简易计算器
 - 1. 运算部分：
 - 2.数值选择器：
 - 3.显示运算结果：
 - 4.整体实现以及PIN映射
 - 2、扩展内容：十进制简易计算器
 - 1.显示模块：
 - 2.整体实现以及PIN映射
 - 3、扩展内容：显示当前运算类型
 - 1.具体实现
- 四、实验中遇到的问题及解决方案
 - 1.基本内容：十六进制简易计算器
 - 2.扩展内容：十进制简易计算器
 - 3.扩展内容：显示当前运算类型

一、实验目的

- 1. 实践基于FPGA设计和实现组合逻辑电路的流程和方法。
- 2. 学习一种硬件描述语言。
- 3. 熟悉利用FPGA平台进行设计验证的方法。

二、实验内容

1.基本内容

基于实验套件中的FPGA开发板，实现如图1所示的简易计算器：

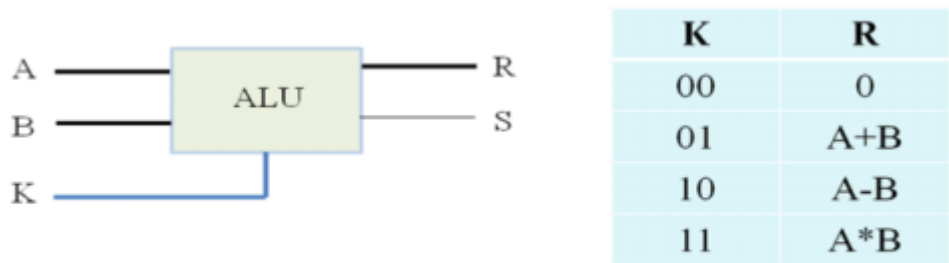


图 1 简易计算器原理图

其中 A 和 B 的取值范围为 0~15；用实验板上的 8 个拨码开关和 2 个按键开关模拟输入 A，B 和 K；通过 4 只数码管显示运算数和运算结果，运算结果的符号用发光二极管来表示。板上 6 只数码管的字段是并接的，通过 6 个选通端控制在哪只数码管上显示；请用 2 个按键开关控制 4 只数码管分别进行显示。图 2 为 FPGA 开发板的示意图。板上的 FPGA 型号为 EP3C16Q240C8。FPGA 引脚与实验箱上的外设的对应关系如图 3 所示。

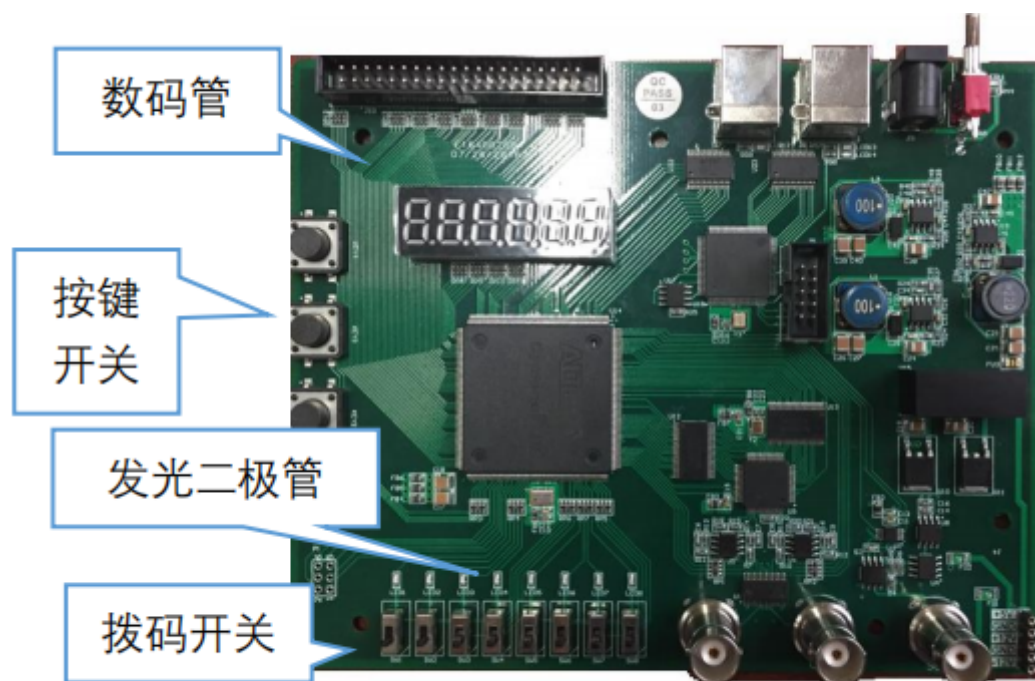
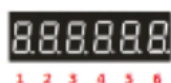


图 2 FPGA 开发板

KEY1	KEY2	KEY3	KEY4
PIN_131	PIN_128	PIN_127	PIN_126
按键按下输出 0, 按键松开输出 1			

SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
PIN_160	PIN_161	PIN_166	PIN_164	PIN_174	PIN_175	PIN_177	PIN_176
开关在上端输出 0, 开关在下端输出 1							

LED1	LED2	LED3	LED4	LED5	LED6	LED7	LED8
PIN_143	PIN_144	PIN_145	PIN_146	PIN_167	PIN_168	PIN_169	PIN_171
输入为 1 时, 相应的发光二极管被点亮							



SEG1	SEG2	SEG3	SEG4	SEG5	SEG6
PIN_69	PIN_64	PIN_57	PIN_51	PIN_50	PIN_49
输入为 0 时, 相应的数码管选通, 可以显示					



A	B	C	D	E	F	G
PIN_56	PIN_68	PIN_65	PIN_55	PIN_70	PIN_52	PIN_63
输入为 0 时, 相应的字段可以被点亮						

图 3 实验板外设引脚

2.进一步研究内容

1. 修改设计, 以十进制形式显示, 要求数码管只显示运算数A和运算结果R;
2. 发挥你的想象, 对简易计算器的功能进行扩展。

三、设计思路

1、基本内容：十六进制简易计算器

1. 运算部分：

- 计算器一共需要实现三种运算：加法、减法、乘法，利用verilog HDL语言可以快速实现。
- 加法器

```
1 module add(a,b,out);
2   input  [3:0] a,b; //a+b
3   output [7:0] out;
4   assign out=a+b;
5 endmodule
```

- 减法器

```
1 module subtract(a,b,out,s);
2   input  [3:0] a,b;          //a-b
3   output [7:0] out;
4   output s;                  //符号位
5   assign s=b>a;              //s=1为负 s=0为正
6   assign out=s?b-a:a-b;
7 endmodule
```

- 乘法器

```
1 module muti(a,b,out);
2   input  [3:0] a,b;
3   output [7:0] out;
4   assign out=a*b;
5 endmodule
```

2.数值选择器：

- 需要根据按钮输入结果，选择计算方式，则需要实现四选一数值选择器；同时有减法会有符号位，需要实现符号位的选择。
- 四选一数值选择器

```
1 module select(i1,i2,i3,k,out);
2   input  [7:0] i1,i2,i3;          //加法 减法 乘法结果
3   input  [1:0] k;                  //按钮输入
4   output [7:0] out;                //输出
5   reg [7:0] out;
6   always @ (i1 or i2 or i3 or k)
7   begin
8       case (k)
9           2'b00: out = 0;          //有一种情况为0，则实际上是只需要三个输入数字即可
10          2'b01: out = i1;
11          2'b10: out = i2;
12          2'b11: out = i3;
13      endcase
14  end
15 endmodule
```

- 符号选择器

```

1 module selectf(s1,s2,s3,k,out);
2 input s1,s2,s3;           //s1 s2 s3 分别为加法 减法 乘法的计算符号，扩展功能考虑
                             实现负数的加法乘法运算，所以先保留了接口
3 input [1:0] k;             //按钮控制
4 output out;
5 reg out;
6 always @ (s1 or s2 or s3 or k)
7 begin
8     case (k)
9         2'b00: out = 1'b0; //当前版本用不到s1 s3，所以用0代替
10        2'b01: out = 1'b0; //
11        2'b10: out = s2;     //负号输出1
12        2'b11: out = 1'b0; //
13    endcase
14 end
15 endmodule

```

3.显示运算结果:

- 因为运算结果需要在并接的数码管上显示，因此需要在输出前进行处理
- 输出处理函数（十六进制）

```

1 module showhex(a,i0,i1,k,dig,seg);
2 input [7:0]a;             //计算结果
3 input [3:0]i0,i1;         //输入数字
4 input [1:0]k;             //显示控制
5 output [3:0]seg;          //控制哪个数码管显示
6 output [6:0]dig;          //数码管显示
7
8 reg [3:0]seg;
9 wire [7:0]num;
10 assign num=a;
11 wire [3:0]b=i0;
12 wire [3:0]c=i1;
13 reg [6:0]out1,out2,out3,out4;
14 reg [6:0]dig;
15
16 always @ (k or a)
17 begin
18     case(k)
19         2'b00: seg=4'b1110; //保证仅有一个数码管可以显示
20         2'b01: seg=4'b1101;
21         2'b10: seg=4'b1011;
22         2'b11: seg=4'b0111;
23     endcase
24 end
25
26 always @ (k or a)
27 begin
28     case(k)                //显示不同的数字
29         2'b00: dig=out1;   //out1:运算结果的高位
30         2'b01: dig=out2;   //out2:运算结果的低位
31         2'b10: dig=out3;   //out3:运算数1
32         2'b11: dig=out4;   //out4:运算数2
33     endcase
34 end

```

```

35
36 always @ (num[7:4])
37 begin
38     case (num[7:4])
39         4'b0000: out1= 7'b0000001; //0
40         4'b0001: out1= 7'b1001111; //1
41         4'b0010: out1= 7'b0010010; //2
42         4'b0011: out1= 7'b0000110; //3
43         4'b0100: out1= 7'b1001100; //4
44         4'b0101: out1= 7'b0100100; //5
45         4'b0110: out1= 7'b0100000; //6
46         4'b0111: out1= 7'b0001111; //7
47         4'b1000: out1= 7'b0000000; //8
48         4'b1001: out1= 7'b0000100; //9
49         4'b1010: out1= 7'b0001000; //A
50         4'b1011: out1= 7'b1100000; //b
51         4'b1100: out1= 7'b0110001; //C
52         4'b1101: out1= 7'b1000010; //D
53         4'b1110: out1= 7'b0110000; //E
54         4'b1111: out1= 7'b0111000; //F
55     endcase
56 end
57
58 always @ (num[3:0])
59 begin
60     case (num[3:0])
61         4'b0000: out2= 7'b0000001;
62         4'b0001: out2= 7'b1001111;
63         4'b0010: out2= 7'b0010010;
64         4'b0011: out2= 7'b0000110;
65         4'b0100: out2= 7'b1001100;
66         4'b0101: out2= 7'b0100100;
67         4'b0110: out2= 7'b0100000;
68         4'b0111: out2= 7'b0001111;
69         4'b1000: out2= 7'b0000000;
70         4'b1001: out2= 7'b0000100;
71         4'b1010: out2= 7'b0001000;
72         4'b1011: out2= 7'b1100000;
73         4'b1100: out2= 7'b0110001;
74         4'b1101: out2= 7'b1000010;
75         4'b1110: out2= 7'b0110000;
76         4'b1111: out2= 7'b0111000;
77     endcase
78 end
79
80 always @ (b)
81 begin
82     case(b[3:0])
83         4'b0000: out3= 7'b0000001;
84         4'b0001: out3= 7'b1001111;
85         4'b0010: out3= 7'b0010010;
86         4'b0011: out3= 7'b0000110;
87         4'b0100: out3= 7'b1001100;
88         4'b0101: out3= 7'b0100100;
89         4'b0110: out3= 7'b0100000;
90         4'b0111: out3= 7'b0001111;
91         4'b1000: out3= 7'b0000000;
92         4'b1001: out3= 7'b0000100;

```

```

93         4'b1010: out3= 7'b0001000;
94         4'b1011: out3= 7'b1100000;
95         4'b1100: out3= 7'b0110001;
96         4'b1101: out3= 7'b1000010;
97         4'b1110: out3= 7'b0110000;
98         4'b1111: out3= 7'b0111000;
99     endcase
100 end
101
102 always @ (c)
103 begin
104     case(c[3:0])
105         4'b0000: out4= 7'b0000001;
106         4'b0001: out4= 7'b1001111;
107         4'b0010: out4= 7'b0010010;
108         4'b0011: out4= 7'b0000110;
109         4'b0100: out4= 7'b1001100;
110         4'b0101: out4= 7'b0100100;
111         4'b0110: out4= 7'b0100000;
112         4'b0111: out4= 7'b0001111;
113         4'b1000: out4= 7'b0000000;
114         4'b1001: out4= 7'b0000100;
115         4'b1010: out4= 7'b0001000;
116         4'b1011: out4= 7'b1100000;
117         4'b1100: out4= 7'b0110001;
118         4'b1101: out4= 7'b1000010;
119         4'b1110: out4= 7'b0110000;
120         4'b1111: out4= 7'b0111000;
121     endcase
122 end
123 endmodule

```

4.整体实现以及PIN映射












- main文件

```

1  module main(i0,i1,k0,k1,seg,dig,sym);
2  input [3:0]i0,i1;          //输入数字1 输入数字2
3  input [1:0]k0,k1;          //两组控制按钮
4  output [3:0]seg;           //决定显示的数码管
5  output [6:0]dig;           //数码管显示数字
6  output sym;                //符号位
7
8  wire [7:0]add_o,sub_o,muti_o,sel_o;
9  wire sub_s;
10 //运算部分
11 add adder(i0,i1,add_o);
12 subtract subtractor(i0,i1,sub_o,sub_s);
13 muti mutier(i0,i1,muti_o);
14 //数值选择
15 select selector(add_o,sub_o,muti_o,k0,sel_o);
16 selectf selectfr(1'b0,sub_s,1'b0,k0,sym);
17 //显示
18 showhex show(sel_o,i0,i1,k1,dig,seg);
19 endmodule

```

- PIN PLANNER

 out	dig[6]	Location	PIN_56
 out	dig[5]	Location	PIN_68
 out	dig[4]	Location	PIN_65
 out	dig[3]	Location	PIN_55
 out	dig[2]	Location	PIN_70
 out	dig[1]	Location	PIN_52
 out	dig[0]	Location	PIN_63
 in	i0[3]	Location	PIN_160
 in	i0[2]	Location	PIN_161
 in	i0[1]	Location	PIN_166
 in	i0[0]	Location	PIN_164
 in	i1[3]	Location	PIN_174
 in	i1[2]	Location	PIN_175
 in	i1[1]	Location	PIN_177
 in	i1[0]	Location	PIN_176
 in	k0[1]	Location	PIN_131
 in	k0[0]	Location	PIN_128
 in	k1[1]	Location	PIN_127
 in	k1[0]	Location	PIN_126
 out	seg[3]	Location	PIN_49
 out	seg[2]	Location	PIN_50
 out	seg[1]	Location	PIN_64
 out	seg[0]	Location	PIN_69
 out	sym	Location	PIN_144

dig[6]到dig[0]对应数码管上的A到G。

i0[3]到i0[0]对应SW1-4， i1[3]到i1[0]对应SW5-8。

k0[1]、[2]对应KEY1、2， k1[1]、[0]对应KEY3、4。按下表示1， 松开表示0。

seg[3]到seg[0]分别对应SEG6、SEG5、SEG2、SEG1。

2、扩展内容：十进制简易计算器

实际上十进制计算器在计算部分与十六进制没有区别，只需要修改显示模块以及主模块即可。

1.显示模块：

```

1  module showhex(clk,a,i0,dig,seg);
2  input  clk;//时钟
3  input  [7:0]  a;//运算结果
4  input  [3:0]  i0;//计算数A
5  output [4:0]  seg;//控制数码管
6  output [6:0]  dig;//数码管显示
7
8  reg [3:0]  n1,n2,n3;
9  reg [3:0]  i1,i2;
10
11 reg [3:0]  nr;
12 reg [4:0]  seg;
13 reg [6:0]  dig;
14 reg [15:0] cnt;//计时器
15

```

```

16 always @ (posedge clk)//计时
17 begin
18     cnt<=cnt+1'b1;
19 end
20
21
22 always @ (a)
23 begin
24     n1=a%10;
25     n2=(a/10)%10;
26     n3=a/100;
27
28 end
29
30 always @ (i0)
31 begin
32     i1=i0%10;
33     i2=i0/10;
34 end
35
36 always @ (cnt[15:13])
37 begin
38     case (cnt[15:13])
39         3'b000: seg <= 5'b11110;//结果第一位
40         3'b001: seg <= 5'b11101;//结果第二位
41         3'b010: seg <= 5'b11011;//结果第三位
42         3'b011: seg <= 5'b10111;//输入第一位
43         3'b100: seg <= 5'b01111;//输入第二位
44         default: seg <=5'b11111;//不要显示
45     endcase
46 end
47
48
49 always @ (n1 or n2 or n3 or i1 or i2 or cnt[15:13])
50 begin
51     case (cnt[15:13])
52         3'b000: nr <= n1;
53         3'b001: nr <= n2;
54         3'b010: nr <= n3;
55         3'b011: nr <= i1;
56         3'b100: nr <= i2;
57         default: nr<=0;
58     endcase
59 end
60
61
62 always @ (nr)
63 begin
64     case (nr)
65         4'b0000: dig<= 7'b0000001;//0
66         4'b0001: dig<= 7'b1001111;//1
67         4'b0010: dig<= 7'b0010010;//2
68         4'b0011: dig<= 7'b0000110;//3
69         4'b0100: dig<= 7'b1001100;//4
70         4'b0101: dig<= 7'b0100100;//5
71         4'b0110: dig<= 7'b0100000;//6
72         4'b0111: dig<= 7'b0001111;//7
73         4'b1000: dig<= 7'b0000000;//8

```



```

74         4'b1001: dig<= 7'b0000100;//9
75         default: dig<= 7'b1111111;//不显示
76     endcase
77 end
78 endmodule

```

- 相对于十六进制的主要改动：
 - 输出三位、输入A两位，一共需要五位数字，因此seg位宽为5，同时dig的显示只需0-9即可，A-F的部分可以不要。
 - 采用扫描显示的模式，利用计时器cnt[15:0]，进行降频处理。
 - 为了避免case条件不完整的情况，当没有用的时候让数码管不显示，最大程度减少数码管之间的干扰。

2.整体实现以及PIN映射

- main文件

```

1  module main(clk,i0,i1,k0,seg,dig,sym);
2  input [3:0]i0,i1;           //输入数字1 输入数字2
3  input [1:0]k0;             //两组控制按钮
4  input clk;
5  output [4:0]seg;           //决定显示的数码管
6  output [6:0]dig;           //数码管显示数字
7  output sym;                //符号位
8
9  wire [7:0]add_o,sub_o,muti_o,sel_o;
10 wire sub_s;
11 //运算部分
12 add adder(i0,i1,add_o);
13 subtract subtractor(i0,i1,sub_o,sub_s);
14 muti mutier(i0,i1,muti_o);
15 //数值选择
16 select selector(add_o,sub_o,muti_o,k0,sel_o);
17 selectf selectfr(1'b0,sub_s,1'b0,k0,sym);
18 //显示
19 showhex show(clk,sel_o,i0,dig,seg);
20 endmodule
21

```

- 相对于十六进制的主要改动：
 - 调整了输入参数，加入了clk输入，同时改变了seg的位宽，调用showhex时调整了参数。
 - （为了验收时方便下载演示，实际上十六进制和十进制的代码是写在同一个文件中的，使用一部分的时候会把另外一部分注释掉，所以showhex模块也没有修改名字，可能会引起歧义！）

- PIN PLANNER

out dig[6]	Location	PIN_56
out dig[5]	Location	PIN_68
out dig[4]	Location	PIN_65
out dig[3]	Location	PIN_55
out dig[2]	Location	PIN_70
out dig[1]	Location	PIN_52
out dig[0]	Location	PIN_63
in i0[3]	Location	PIN_160
in i0[2]	Location	PIN_161
in i0[1]	Location	PIN_166
in i0[0]	Location	PIN_164
in i1[3]	Location	PIN_174
in i1[2]	Location	PIN_175
in i1[1]	Location	PIN_177
in i1[0]	Location	PIN_176
in k0[1]	Location	PIN_131
in k0[0]	Location	PIN_128
? k1[1]	Location	PIN_127
? k1[0]	Location	PIN_126
out seg[3]	Location	PIN_50
out seg[2]	Location	PIN_69
out seg[1]	Location	PIN_64
out seg[0]	Location	PIN_57
out sym	Location	PIN_144
in clk	Location	PIN_152
out seg[4]	Location	PIN_51

输入映射没有调整，在输出时做了调整。

输出对应数码管最左面三个，从左到右依次是高位到低位。

右面两个是输入数字A的数码管，左面高位，右面低位。

3、扩展内容：显示当前运算类型

在前面做实验的过程中，为了检查实现是否正确，需要对于每种运算的每个情况进行检查，虽然能够看见两个输入数字，但是不能简单区分目前在做的是**哪种运算**。所以在做完十进制计算器后，利用最后剩下的一个数码管显示了当前的运算。

1.具体实现

- 非常简单，在显示模块的输入参量中加入控制按钮k0[1:0]，并且在扫描显示时将这个也作为一个显示数据加入。同时将seg变成[5:0]，再调整一下main文件中的seg就完成了。
- **显示模块代码：**（跟上面那个区别不大，同时由于main文件区别实在太小，就不放进来了）

```

1 module showhex(clk,a,i0,k0,dig,seg);
2   input clk;//时钟
3   input [7:0] a;//运算结果
4   input [3:0] i0;//计算数A
5   input [1:0] k0;//控制按钮
6   output [5:0] seg;//控制数码管
7   output [6:0] dig;//数码管显示
8
9   reg [3:0] n1,n2,n3;
```

```

10 reg [3:0] i1,i2;
11
12 reg [3:0] nr;
13 reg [5:0] seg;
14 reg [6:0] dig;
15 reg [15:0] cnt;
16 reg [4:0] kr;
17
18 always @ (posedge clk)//计时
19 begin
20     cnt<=cnt+1'b1;
21 end
22
23
24 always @ (a)
25 begin
26     n1=a%10;
27     n2=(a/10)%10;
28     n3=a/100;
29
30 end
31
32 always @ (i0)
33 begin
34     i1=i0%10;
35     i2=i0/10;
36 end
37
38
39 always @ (k0)
40 begin
41     kr[1:0]=k0;
42 end
43
44
45 always @ (cnt[15:13])
46 begin
47     case (cnt[15:13])
48         3'b000: seg <= 6'b111110;//结果第一位
49         3'b001: seg <= 6'b111101;//结果第二位
50         3'b010: seg <= 6'b111011;//结果第三位
51         3'b011: seg <= 6'b110111;//输入第一位
52         3'b100: seg <= 6'b101111;//输入第二位
53         3'b101: seg <= 6'b011111;
54         default: seg <=6'b111111;//不要显示
55     endcase
56 end
57
58
59 always @ (n1 or n2 or n3 or i1 or i2 or kr or cnt[15:13])
60 begin
61     case (cnt[15:13])
62         3'b000: nr <= n1;
63         3'b001: nr <= n2;
64         3'b010: nr <= n3;
65         3'b011: nr <= i1;
66         3'b100: nr <= i2;
67         3'b101: nr <= kr;

```

```

68         default: nr<=0;
69     endcase
70 end
71
72
73 always @ (nr)
74 begin
75     case (nr)
76         4'b0000: dig<= 7'b0000001;//0
77         4'b0001: dig<= 7'b1001111;//1
78         4'b0010: dig<= 7'b0010010;//2
79         4'b0011: dig<= 7'b0000110;//3
80         4'b0100: dig<= 7'b1001100;//4
81         4'b0101: dig<= 7'b0100100;//5
82         4'b0110: dig<= 7'b0100000;//6
83         4'b0111: dig<= 7'b0001111;//7
84         4'b1000: dig<= 7'b0000000;//8
85         4'b1001: dig<= 7'b0000100;//9
86         default: dig<= 7'b1111111;//不显示
87     endcase
88 end
89 endmodule

```

四、实验中遇到的问题及解决方案

1.基本内容：十六进制简易计算器

1. 在上手verilog语言时，对于基本的语法规则并不熟悉，经常编译出错。
 - 解决：上网查阅文档，熟悉基本写法，以及对error进行检索，了解原因。
2. 如何保证每个模块的正确性以及调试
 - 解决：在设计按功能将代码实现分为几个部分，尽量保证功能间的去耦合。同时每个部分设计完毕后都进行波形模拟，检查正确性。
3. 第一次将程序下载到板子上后发现出现问题
 - 解决：发现主要错误在于控制哪个数码管显示的按钮工作不正常，重新检查了一下显示模块波形，发现原因在于always语句的敏感变量不全，并且在赋值时把b打成了d。

2.扩展内容：十进制简易计算器

1. 在加入cnt参量时，发现位数过低的话数码管显示不稳定
 - 解决：最初用的是7-5位，后来调整成了15-13位。
2. 下载到板子上调试时，发现显示不正常，经检查发现是main函数的输入参量seg未调整位宽适应。
 - 解决：重新调整了输入的位宽

3.扩展内容：显示当前运算类型

这个功能实现起来实在是很快，5分钟就搞完了，也没遇到任何问题。

