111-1　535313

進階可程式邏輯系統設計與應用

Advanced Programmable Logic System Design and Application

實驗編號 : LAB 01

實驗名稱 : Add two 1-digit BCD numbers

結果報完成日期 : 2022.10.16

姓名 : 王語

系級 : 機械四

學號 : 0811127

一、 實驗目的

藉由全加器組成連波加法器、多位元多工器、七段顯示器解碼器、溢位判斷等等設計 BDC 加法器，並將輸入、計算結果顯示於七段顯示器，當輸入超過 9 時 LEDR 亮起偵錯，也藉由實驗熟悉 Quartus 軟體之使用。

二、 Verilog 程式碼

總電路連接結果可以直接看 RTL 會更淺顯易懂。

Lab_01_0811127.v

```
1   // 0811127 王語
2
3   // Top Module
4   module Lab_01_0811127 ( input  [8:0] SW,
5                           output [9:9] LEDR,
6                           output [6:0] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5 );
7   wire errtemp1 , errtemp2 ;
8
9   wire [3:0] sum;
10  wire cout;
11
12  BCD_converter no_name_haha1 (   .v(SW[3:0]),
13                                  .d1(HEX1),
14                                  .d0(HEX0),
15                                  .v5(0),
16                                  .err(errtemp1) );
17
18  BCD_converter no_name_haha2 (   .v(SW[7:4]),
19                                  .d1(HEX3),
20                                  .d0(HEX2),
21                                  .v5(0),
22                                  .err(errtemp2) );
23
24  BCD_converter no_name_haha3 (   .v(sum),
25                                  .d1(HEX5),
26                                  .d0(HEX4),
27                                  .v5(cout),
28                                  .err() ); //err 忽略
29
30  fourbits_adder fourbitsFFD (.a(SW[3:0]),
31                              .b(SW[7:4]),
32                              .Cin(SW[8]),
33                              .Cout(cout),
34                              .sum(sum) );
35
36  assign LEDR[9] = errtemp1|errtemp2;
37
38  endmodule
```

```
39
40  //Submodules
41  module BCD_converter (  input [3:0] v,
42                          input v5 , // 第五個bit
43                          output [6:0] d1 , d0 ,
44                          output err );
45  wire z ;
46  wire [3:0] vps; //v plus six , 加六進位
47  wire [6:0] dtemp;
48  // integer six = 4'd6;
49  //(X) assign six = 4'd6 ;
50
51  fourbits_adder fourbitsFFD (.a(v),
52                              .b(4'b0110),
53                              .Cin(0),
54                              .Cout(),
55                              .sum(vps) );
56
57  fourbits_sel Fourbits_sel(  .s(z|v5),
58                              .a(vps),
59                              .b(v),
60                              .out(dtemp) ); // 用sel 取代 if (z) ... else...
61
62  comparator com_1 (  .v(v),
63                      .z(z) );
64
65
66  assign err = z ;
67  assign d1[1] = ~(z|v5) ;
68  assign d1[2] = ~(z|v5) ; // 最多19 直接給1 不然就 0
69  assign d1 [0] = 1;
70  //assign d1 [3] = 0;
71  assign d1[6:3] = 4'b1111; // 共陽極
72  ssd seven_segment_decoder ( .Din(dtemp),
73                              .Dout(d0) );
74
75
76  endmodule
```

```
78  module comparator ( input [3:0] v,
79                      output z );
80
81  assign z = (v[3]&v[2])|(v[1]&v[3]); // 真直表得出
82
83  endmodule
84
85  module sel (input a, b, s,
86              output out );    //這邊變數改成A0 A1 會更好判讀
87  assign out = (a&s)|(b&(~s)); //s=0 >> b , s=1 >> a
88  endmodule
89
90  module fourbits_sel (   input [3:0] a, b,
91                          input s,
92                          output [3:0] out );
93  //generate 真香
94  sel sel0(   .a(a[0]),
95              .b(b[0]),
96              .s(s),
97              .out(out[0]) );
98
99  sel sel1(   .a(a[1]),
100             .b(b[1]),
101             .s(s),
102             .out(out[1]) );
103
104 sel sel2(   .a(a[2]),
105             .b(b[2]),
106             .s(s),
107             .out(out[2]) );
108
109 sel sel3(   .a(a[3]),
110             .b(b[3]),
111             .s(s),
112             .out(out[3]) );
113
114 endmodule
```

Z 用來判斷輸入數值 A 和 B 是否超過 10

```
//七段顯示器
module ssd (    input [3:0] Din,
                output [6:0] Dout );
assign Dout[0] =    ((!Din[3])&(!Din[2])&(!Din[1])&( Din[0]))|
                    ((!Din[3])&( Din[2])&(!Din[1])&(!Din[0]))|
                    (( Din[3])&( Din[2])&(!Din[1])&( Din[0]))|
                    (( Din[3])&(!Din[2])&( Din[1])&( Din[0]));

assign Dout[1] =    ((!Din[3])&( Din[2])&(!Din[1])&( Din[0]))|
                    (          ( Din[2])&( Din[1])&(!Din[0]))|
                    (( Din[3])&          ( Din[1])&( Din[0]))|
                    (( Din[3])&( Din[2])&          (!Din[0]));

assign Dout[2] =    ((!Din[3])&(!Din[2])&( Din[1])&(!Din[0]))|
                    (( Din[3])&( Din[2])&( Din[1])          )|
                    (( Din[3])&( Din[2])&          (!Din[0]));


assign Dout[3] =    ((!Din[3])&(!Din[2])&(!Din[1])&( Din[0]))|
                    ((!Din[3])&( Din[2])&(!Din[1])&(!Din[0]))|
                    (( Din[3])&(!Din[2])&( Din[1])&(!Din[0]))|
                    (          ( Din[2])&( Din[1])&( Din[0]));

assign Dout[4] =    ((!Din[3])&                  &( Din[0]))|
                    (          (!Din[2])&(!Din[1])&( Din[0]))|
                    ((!Din[3])&( Din[2])&(!Din[1])          );


assign Dout[5] =    (( Din[3])&( Din[2])&(!Din[1])&( Din[0]))|
                    ((!Din[3])&(!Din[2])&          ( Din[0]))|
                    ((!Din[3])&(!Din[2])&( Din[1])          )|
                    ((!Din[3])&          ( Din[1])&( Din[0]));

assign Dout[6] =    ((!Din[3])&( Din[2])&( Din[1])&( Din[0]))|
                    (( Din[3])&( Din[2])&(!Din[1])&(!Din[0]))|
                    ((!Din[3])&(!Din[2])&(!Din[1])          );

endmodule
```

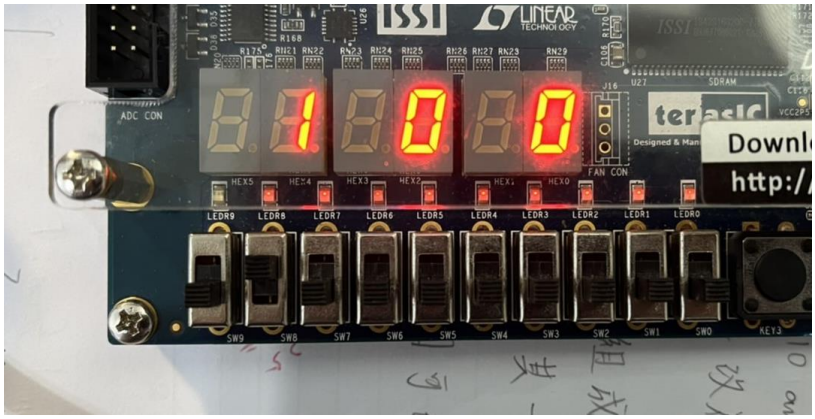
這邊就是使用 kmap 化減工具輔助我得出結果。

```
155  //element of fulladder
156  module efulladder ( input cin , a0 , a1 ,
157                      output cout , sum );
158  wire [2:0] temp0 ,temp1;
159  assign temp0 = {cin,a0,a1};
160  assign sum = ^temp0[2:0];
161  assign temp1 = {(a0&a1),(a1&cin),(a0&cin)};
162  assign cout = |temp1;
163
164  endmodule
165
166  //ripple adder
167  module fourbits_adder ( input [3:0] a,b,
168                          input Cin,
169                          output Cout,
170                          output [3:0] sum );
171
172  wire [3:0] cout;
173  // 用generate可以省下很多時間
174  efulladder fad0(.cin(Cin),
175                  .a0(a[0]),
176                  .a1(b[0]),
177                  .cout(cout[0]),
178                  .sum(sum[0]) );
179  efulladder fad1(.cin(cout[0]),
180                  .a0(a[1]),
181                  .a1(b[1]),
182                  .cout(cout[1]),
183                  .sum(sum[1]) );
184  efulladder fad2(.cin(cout[1]),
185                  .a0(a[2]),
186                  .a1(b[2]),
187                  .cout(cout[2]),
188                  .sum(sum[2]) );
189  efulladder fad3(.cin(cout[2]),
190                  .a0(a[3]),
191                  .a1(b[3]),
192                  .cout(cout[3]),
193                  .sum(sum[3]) );
194
195  assign Cout = cout[3];
196
197  endmodule
```

三、　實驗結果

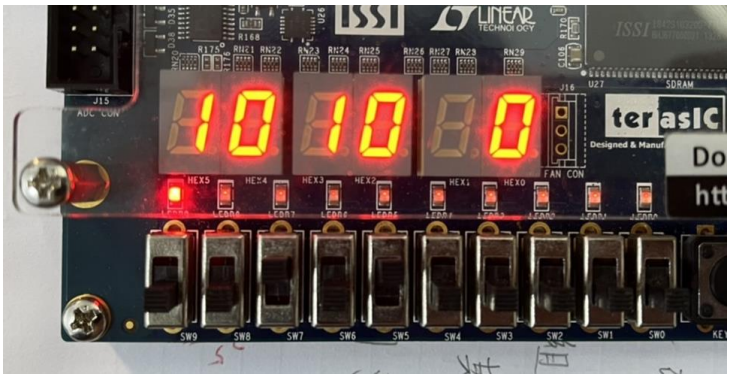
初始狀態 : 0 + 0 + 0 = 0

Cin = 1

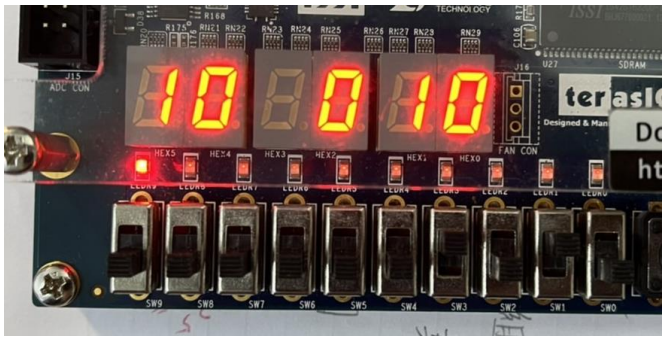


Cin = 1 , A = 9 , B = 9

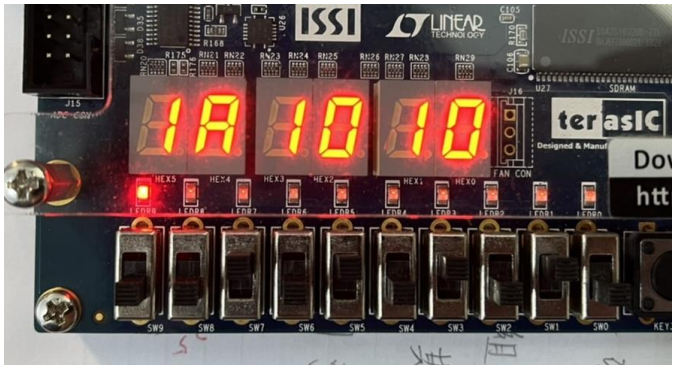
Cin = 0 , B = 10 , A = 0



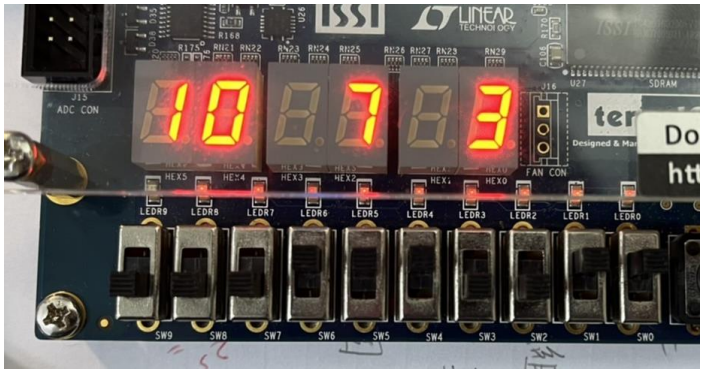
Cin = 0 , A = 10 , B = 10



A = 10 , B = 10

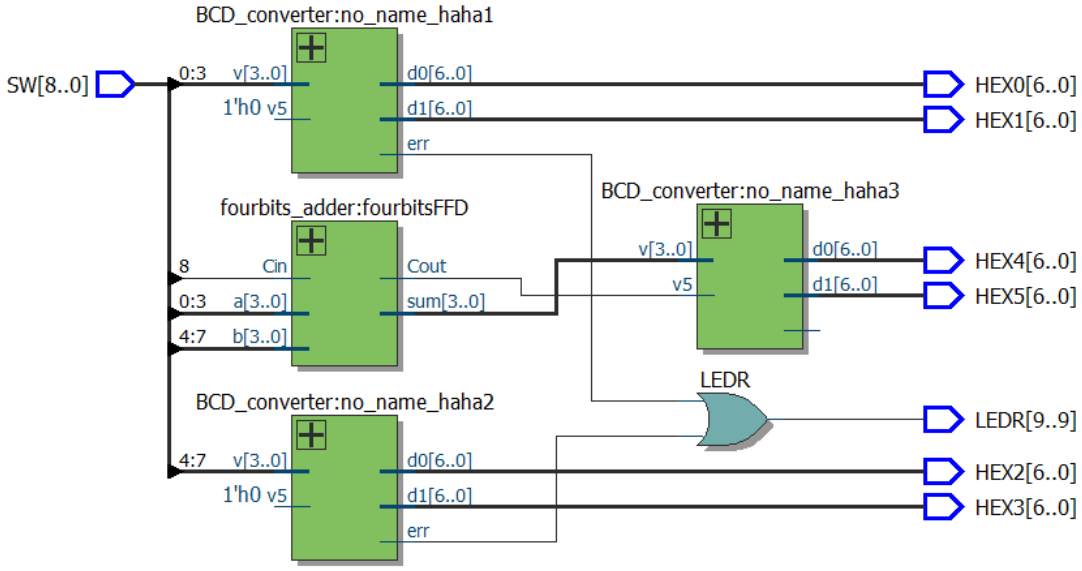


A = 3 , B = 7



四、 RTL

總電路



完整版連結 ： https://pse.is/4jg36w

五、 問題與討論

1. 在七段顯示器真直表化減過程中花費大量時間，才化減出其中一個，我認為太沒效率，因此在網路上找到 kmap 化減工具，真的很好用，獲得大幅加速。
2. 因為看不習慣 Quartus 開發環境的字體以及顏色，花了一些時間找到整合 VScode 編輯環境，好讓我能使用 VScode 寫 Verilog 程式，並在 Quartus II 下完成 compile、Pin plan、燒錄、RTL 輸出等作業。
3. 在開始設計電路前有先草擬一下架構，BCD_converter 一開始只有 4bits 的輸入([3:0] v)，但是這樣最多只能呈現 0~15 的情況，因此緊夾增加 v5，好在這個疏失並沒有太多要調整的地方，是否有十位數只要判斷是否超過 10(可由 z 決定)或是否有第五位數字(v5 = 1)，因此只要把

```
assign d1[1] = ~(z) ;
assign d1[2] = ~(z) ;
```

改成

```
assign d1[1] = ~(z|v5) ;
assign d1[2] = ~(z|v5) ;
```

4. 因為沒有 if else 可以使用，因此在設計時用 sel 替代這項功能。

六、 心得

這次實驗收穫滿滿，複習了基本的電路設計，在沒有 always 可以用的前提下，真的很需要動腦，不過設計電路這種東西本來就是要從最基本的開始吧，基礎先穩固才能學更厲害的東西，感謝助教批閱。

七、 備註

助教不好意思我有一些 module 亂命名，希望您不要介意，下次會好好取名字。