FPGA 實驗一

Add two 1-digit BCD numbers

學生: 謝旻錡

學號: 313512078

日期: 2025/03/01

1. 實驗目的

主要目標:

為藉由撰寫 Add two 1-digit numbers BCD 來熟悉 Verilog 編程、熟悉開發版及開發環境。

實驗流程:

實驗 part3 練習用 switch 及 LED 和邏輯電路組成 multiplexers。

實驗 part4 練習使用 7-segment display 及複習邏輯設計的卡諾圖(4 變數)。

實驗 part5 基於 part4 將其從 0-9 的顯示器 (1-digit)擴展至 0-15 的 BCD (binary coded decimal) 顯示器 (4-bit to 2-digit)。

實驗 part6 配合 part3 的 multiplexers 練習 Full Adder 的邏輯電路設計，並將 4 個 Full Adders 串接組合成 4bit + 4bit = 5bit Four-bit ripple-carry Adder。

最終 Demo，實驗 part7 實作 Add two 1-digit numbers BCD，提供一個以 switch 輸入的 4bit + 4bit = 5bit 的 BCD 顯示器。其中，會使用到實驗 part5、part6 的結果和一個 5-bit 的 BCD。

2. 實驗程式碼

Demo part7: Add two 1-digit BCD numbers 程式碼，其餘 part2~part6 程式碼放在附錄。

Add two 1-digit BCD numbers 是由 一個 Four-bit ripple-carry Adder、兩個 4-bit BCD 和一個 5-bit BCD 組合成的邏輯電路。

計算時我使用 http://www.32x8.com/ online Logic circuit simplification (SOP and POS) 及 Chatgpt 輔助簡化邏輯計算。

/*part7: Add two 1-digit BCD numbers*/

```
1.  module lab1(SW, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, LEDR);
2.  input [9:0] SW;
3.  output [9:0] LEDR;
4.  output [6:0]    HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
5.
6.  wire Cin, Cout;
7.  wire[3:0] A, B, S;
8.  wire [3:1] C;
9.  //Full Adder
10. assign A = SW [7:4];
11. assign B = SW [3:0];
12. assign Cin = SW[8];
13.
14. assign S[0] = ((Cin)^(A[0] ^ B[0]));
15. assign C[1] = (~(A[0] ^ B[0]) & B[0]) | ((A[0] ^ B[0]) & Cin);
16. assign S[1] = ((C[1])^(A[1] ^ B[1]));
17. assign C[2] = (~(A[1] ^ B[1]) & B[1]) | ((A[1] ^ B[1]) & C[1]);
18. assign S[2] = ((C[2])^(A[2] ^ B[2]));
19. assign C[3] = (~(A[2] ^ B[2]) & B[2]) | ((A[2] ^ B[2]) & C[2]);
```

```
20. assign S[3] = ((C[3])^(A[3] ^ B[3]));
21. assign Cout = (~(A[3] ^ B[3]) & B[3]) | ((A[3] ^ B[3]) & C[3]);
22. //BCD
23. //HEX1~0
24. wire a,b,c,d;
25. assign a = SW[3];
26. assign b = SW[2];
27. assign c = SW[1];
28. assign d = SW[0];
29.
30. assign HEX0[6] = ~((b && ~c) || (b &&~d) || (a && ~c) || (a && b) || (~a
    && ~b && c));
31. assign HEX0[5] = ~((~a && ~c && ~d) || (~a && b && ~c) || (~a && b
    && ~d) || (a && ~b && ~c) || (a && ~b && ~d) || (a && b && c));
32. assign HEX0[4] = ~((~b && ~d) || (~a && c && ~d) || (a && ~c && ~d));
33. assign HEX0[3] = ~((~b && ~d) || (a && ~c) || (~a && ~b && c) || (~a
    && c && ~d) || (b && ~c && d) || (a && b && d));
34. assign HEX0[2] = ~((d) || (~a && ~c) || (b && c) || (a && ~b));
35. assign HEX0[1] = ~((~b) || (~c && ~d) || (a &&~c) || (a &&~d) || (~a &&
    c && d));
36. assign HEX0[0] = ~((~b && ~d) || (~a && c) || (b && d) || (a && ~c));
37.
38. assign HEX1[6] = ~(0);
39. assign HEX1[5] = ~((~a) || (~b && ~c));
40. assign HEX1[4] = ~((~a) || (~b && ~c));
41. assign HEX1[3] = ~((~a) || (~b && ~c));
42. assign HEX1[2] = ~(1);
```

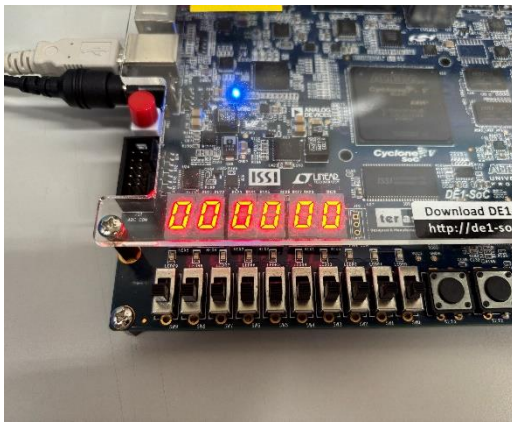
```
43. assign HEX1[1] = ~(1);
44. assign HEX1[0] = ~((~a) || (~b && ~c));
45.
46. //HEX3~2
47. wire a2,b2,c2,d2;
48. assign a2 = SW[7];
49. assign b2 = SW[6];
50. assign c2 = SW[5];
51. assign d2 = SW[4];
52.
53. assign HEX2[6] = ~((b2 && ~c2) || (b2 && ~d2) || (a2 && ~c2) || (a2 &&
    b2) || (~a2 && ~b2 && c2));
54. assign HEX2[5] = ~((~a2 && ~c2 && ~d2) || (~a2 && b2 && ~c2) || (~a2
    && b2 && ~d2) || (a2 && ~b2 && ~c2) || (a2 && ~b2 && ~d2) || (a2 &&
    b2 && c2));
55. assign HEX2[4] = ~((~b2 && ~d2) || (~a2 && c2 && ~d2) || (a2 && ~c2
    && ~d2));
56. assign HEX2[3] = ~((~b2 && ~d2) || (a2 && ~c2) || (~a2 && ~b2 && c2)
    || (~a2 && c2 && ~d2) || (b2 && ~c2 && d2) || (a2 && b2 && d2));
57. assign HEX2[2] = ~((d2) || (~a2 && ~c2) || (b2 && c2) || (a2 && ~b2));
58. assign HEX2[1] = ~((~b2) || (~c2 && ~d2) || (a2 &&~c2) || (a2 &&~d2) ||
    (~a2 && c2 && d2));
59. assign HEX2[0] = ~((~b2 && ~d2) || (~a2 && c2) || (b2 && d2) || (a2 &&
    ~c2));
60.
61. assign HEX3[6] = ~(0);
62. assign HEX3[5] = ~((~a2) || (~b2 && ~c2));
63. assign HEX3[4] = ~((~a2) || (~b2 && ~c2));
```

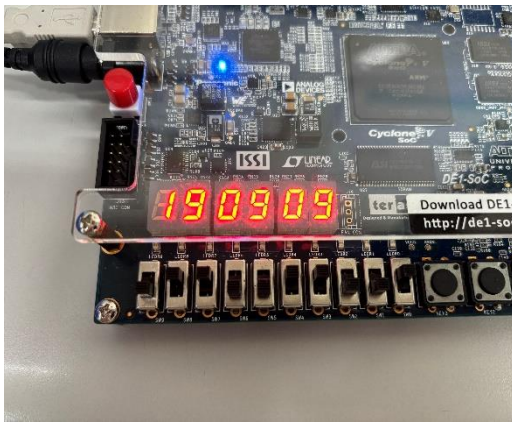
```
64. assign HEX3[3] = ~((~a2) || (~b2 && ~c2));
65. assign HEX3[2] = ~(1);
66. assign HEX3[1] = ~(1);
67. assign HEX3[0] = ~((~a2) || (~b2 && ~c2));
68.
69. //HEX5~4 5bit-BCD
70. wire a3,b3,c3,d3;
71. assign a3 = Cout;
72. assign b3 = S[3];
73. assign c3 = S[2];
74. assign d3 = S[1];
75. assign e3 = S[0];
76.
77. assign HEX4[6] = ~((b3 & ~d3) | (~b3 & ~c3 & d3) | (~b3 & d3 & ~e3) |
    (~a3 & c3 & ~d3) | (~a3 & b3 & c3) | (a3 & ~c3 & ~e3) | (a3 & ~b3 &
    d3));
78. assign HEX4[5] = ~((~b3 & ~d3 & ~e3) | (b3 & ~c3 & ~d3) | (b3 & d3 &
    ~e3) | (a3 & b3 & ~d3) | (~a3 & ~b3 & c3 & ~d3) | (~a3 & ~b3 & c3 &
    ~e3) | (~a3 & b3 & c3 & d3) | (a3 & ~b3 & ~c3 & d3));
79. assign HEX4[4] = ~((~b3 & ~c3 & ~e3) | (~b3 & d3 & ~e3) | (~c3 & d3 &
    ~e3) | (a3 & c3 & ~e3) | (~a3 & b3 & ~d3 & ~e3));
80. assign HEX4[3] = ~((~a3 & ~c3 & ~e3) | (~b3 & ~c3 & d3) | (~b3 & d3 &
    ~e3) | (b3 & ~d3 & e3) | (b3 & c3 & ~d3) | (a3 & ~b3 & ~e3) | (a3 & ~b3
    & d3) | (a3 & d3 & ~e3) | (~a3 & c3 & ~d3 & e3) | (~a3 & b3 & c3 & e3));
81. assign HEX4[2] = ~(e3 | (~c3 & ~d3) | (b3 & d3) | (a3 & ~c3) | (a3 & ~d3)
    | (~a3 & ~b3 & c3));
82. assign HEX4[1] = ~((a3 & c3) | (~a3 & ~d3 & ~e3) | (~b3 & ~c3 & e3) |
    (~b3 & ~c3 & d3) | (~b3 & d3 & e3) |(~c3 & d3 & e3) | (~a3 & b3 & ~d3)
    | (~a3 & b3 & ~e3) | (b3 & ~d3 & ~e3));
```

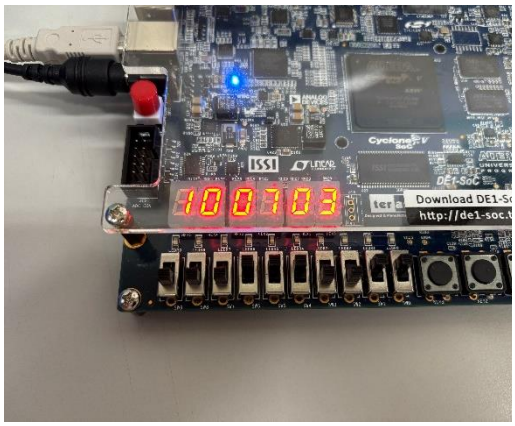
```
83. assign HEX4[0] = ~((~b3 & d3) | (~a3 & ~c3 & ~e3) | (~a3 & c3 & e3) |
    (~a3 & b3 & ~d3) | (b3 & c3 & ~d3) | (a3 & ~b3 & ~e3) | (a3 & ~c3 & e3)
    | (a3 & d3 & ~e3));
84.
85. assign HEX5[6] = ~((a3 & c3) | (a3 & b3));
86. assign HEX5[5] = ~((~a3 & ~b3) | (~a3 & ~c3 & ~d3));
87. assign HEX5[4] = ~((~a3 & ~b3) | (~b3 & c3) | (~a3 & ~c3 & ~d3) | (a3 &
    c3 & ~d3) | (a3 & b3 & ~c3));
88. assign HEX5[3] = ~((~a3 & ~b3) | (~b3 & c3) | (a3 & b3) | (~a3 & ~c3 &
    ~d3));
89. assign HEX5[2] = ~(~a3 | (~b3 & ~c3) | (b3 & c3 & d3));
90. assign HEX5[1] = ~(1);
91. assign HEX5[0] = ~((~a3 & ~b3) | (~b3 & c3) | (a3 & b3) | (~a3 & ~c3 &
    ~d3));
92.
93. //ERROR LEDR[9]
94. assign LEDR[9] = ((a & c) | (a & b)) | ((a2 & c2) | (a2 & b2));
95. endmodule
```

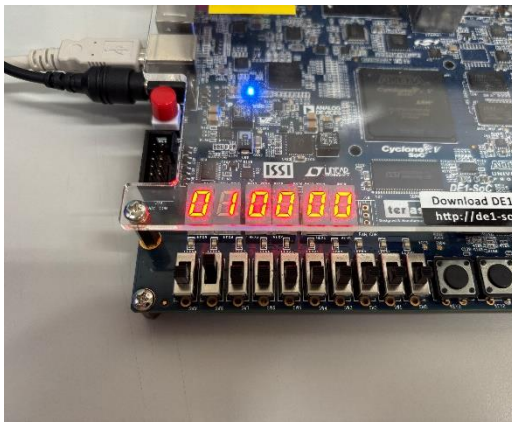
3. 實驗結果照片(optional)

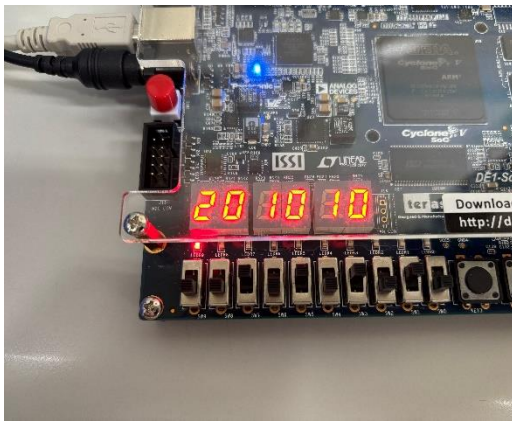
實驗 part7 實作 Add two 1-digit numbers BCD





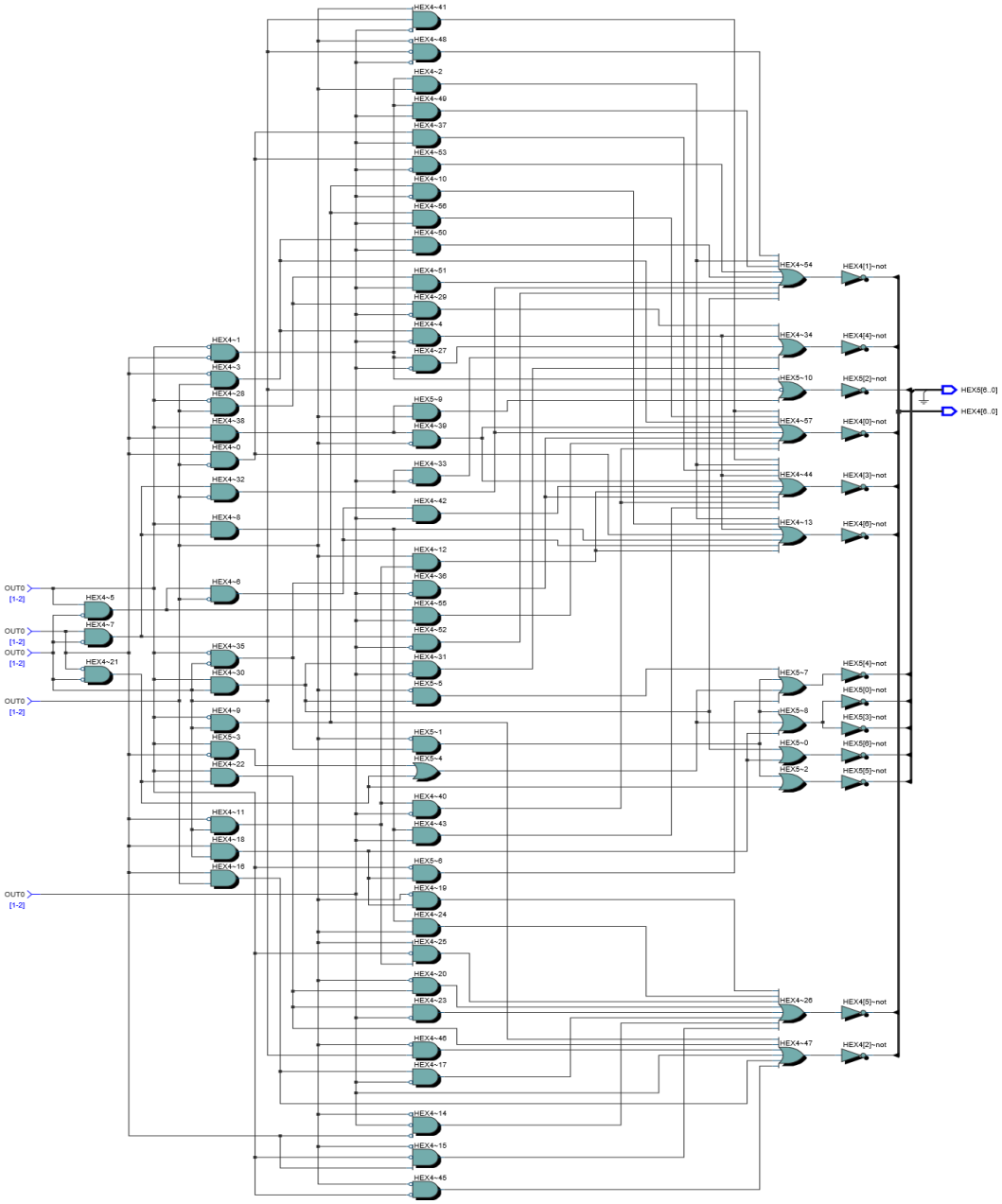


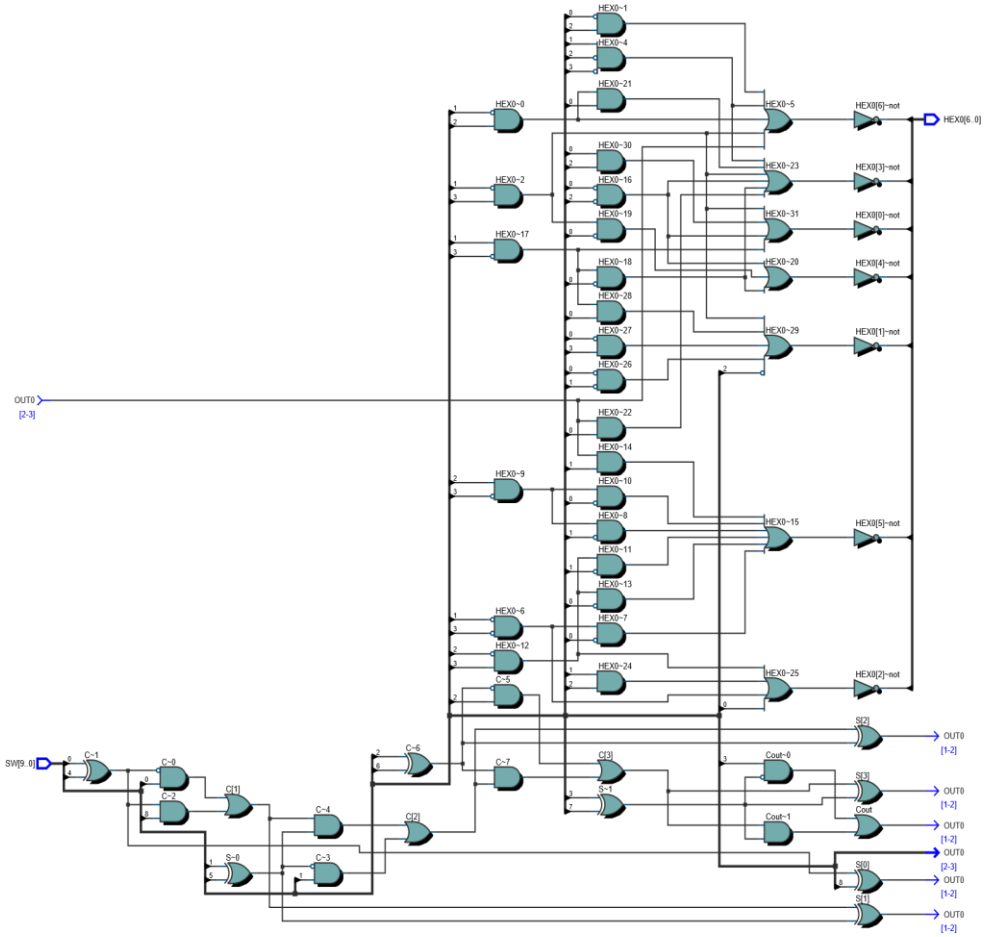


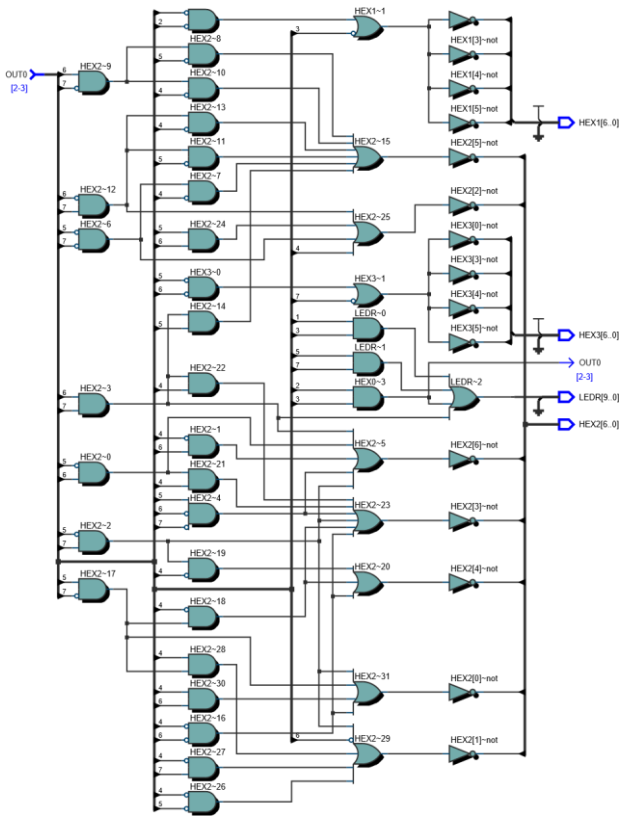


4. RTL 布局(optional)

Tools >> Netlist viewer >> RTL viewer.




HEX0~1
HEX0~4
HEX0~21
HEX0~0
HEX0~5
HEX0[6]~not
HEX0[6..0]
HEX0~30
HEX0~2
HEX0~16
HEX0~23
HEX0[3]~not
HEX0~19
HEX0~31
HEX0[0]~not
HEX0~17
HEX0~18
HEX0~20
HEX0[4]~not
HEX0~28
HEX0~27
HEX0~29
HEX0[1]~not
OUT0
[2-3]
HEX0~26
HEX0~22
HEX0~14
HEX0~9
HEX0~10
HEX0~8
HEX0~15
HEX0[5]~not
HEX0~11
HEX0~13
HEX0~6
HEX0~7
HEX0~12
HEX0~24
HEX0~25
HEX0[2]~not
C~5
SW[9..0]
C~1
C~0
C~6
C~7
C[3]
Cout~0
S[2]
OUT0
[1-2]
C~2
C[1]
S~1
S[3]
OUT0
[1-2]
C~4
C[2]
Cout~1
Cout
OUT0
[1-2]
S~0
C~3
OUT0
[2-3]
S[0]
OUT0
[1-2]
S[1]
OUT0
[1-2]


HEX1~1
HEX1[3]~not
HEX1[4]~not
HEX1[5]~not
HEX1[6..0]
OUT0
[2-3]
HEX2~8
HEX2~9
HEX2~10
HEX2~13
HEX2~11
HEX2~15
HEX2[5]~not
HEX2~7
HEX2~12
HEX2~25
HEX2[2]~not
HEX2~6
HEX2~24
HEX3[0]~not
HEX3~0
HEX3~1
HEX3[3]~not
HEX2~14
HEX3[4]~not
LEDR~0
HEX3[5]~not
HEX3[6..0]
LEDR~1
OUT0
[2-3]
HEX2~22
HEX0~3
LEDR~2
LEDR[9..0]
HEX2~3
HEX2[6]~not
HEX2~1
HEX2~5
HEX2~9
HEX2~21
HEX2~23
HEX2[3]~not
HEX2~4
HEX2~2
HEX2~20
HEX2[4]~not
HEX2~19
HEX2~17
HEX2~18
HEX2~28
HEX2~31
HEX2[0]~not
HEX2~30
HEX2~16
HEX2~29
HEX2[1]~not
HEX2~27
HEX2~26

5. 問題與討論

- **Problem:** 在實驗中，最初對 LED 宣告為 output [3:0] LEDR，燒入後 LED[9:4] 在沒有宣告的狀況下會呈現亮一半的狀態。

  Solution: 在宣告改為 output [9:0] LEDR，可解決問題。

- **Problem:** 在實驗時發現 7-segment display 顯示方式為負片的形式。

  Solution: 原因為 7-segment display 開發版上為共陽極，因此在邏輯前須加上 ~() 反計算式修正邏輯。

- 注意事項:

  1. Compile 前，一定要先確定有 import pin assignment file，這樣才能在燒入時正確連接 pin 腳位。
  2. 如果使用 Chatgpt 將最簡邏輯式傳換成 verliog 函式，續確實檢查，有時會遺漏 ~()。

6. 附錄:

```
/*part2*/
1.  module lab1 (SW, LEDR);
2.  input [17:0] SW;
3.  output [17:0] LEDR;
4.  assign LEDR = SW;
5.  endmodule
```

```
/*part3: 2-1 multiplexer*/
1.  module lab1 (SW,LEDR);
2.  input [9:0] SW;
3.  output [9:0] LEDR;
4.  assign LEDR[3] = (~SW[9] & SW[3]) | (SW[9] & SW[7]);
5.  assign LEDR[2] = (~SW[9] & SW[2]) | (SW[9] & SW[6]);
6.  assign LEDR[1] = (~SW[9] & SW[1]) | (SW[9] & SW[5]);
7.  assign LEDR[0] = (~SW[9] & SW[0]) | (SW[9] & SW[4]);
8.  endmodule
9.  /*problem: LEDR9~4 light half??
10. sol: output [3:0] LEDR =>output [9:0] LEDR*/
```

```
/*part4: 7-segment display*/
1.  module lab1 (SW,HEX0);
2.  input [3:0] SW;
3.  output [6:0] HEX0;
4.
5.  assign a = SW[3];
```

```
6.  assign b = SW[2];
7.  assign c = SW[1];
8.  assign d = SW[0];
9.  //HEX is common anode configuration
10. assign HEX0[6] = ~((~a && b && ~c) || (a && ~b &&~c) || (~a && ~b &&c) || (~a && c && ~d));
11. assign HEX0[5] = ~((~a && ~c && ~d) || (~a && b && ~c) ||(a &&~b && ~c) ||(~a && b && c && ~d));
12. assign HEX0[4] = ~((~b && ~c && ~d) || (~a && c &&~d));
13. assign HEX0[3] = ~((~a && ~b && ~c && ~d) || (~a && b && ~c && d) || (a && ~b && ~c) || (~a && ~b && c) || (~a && c && ~d));
14. assign HEX0[2] = ~((~a && ~b && ~c) || (~a && b) || (a && ~b && ~c) || (~a && c && d));
15. assign HEX0[1] = ~((~a && ~b) || (a && ~b && ~c) || (~a && ~c && ~d) || (~a && c && d));
16. assign HEX0[0] = ~((~a && ~b && ~c && ~d) || (a && ~b && ~c) || (~a && b && d) || (~a && c && ~d ) || (~a && ~b && c));
17. endmodule
```

```
/*part5: BCD binary coded decimal*/
1.  module lab1 (SW,HEX1,HEX0);
2.  input [3:0] SW;
3.  output [6:0] HEX1, HEX0;
4.
5.  assign a = SW[3];
6.  assign b = SW[2];
7.  assign c = SW[1];
```

```
8.  assign d = SW[0];
9.  //HEX is common anode configuration
10. assign HEX0[6] = ~((b && ~c) || (b &&~d) || (a && ~c) || (a && b) || (~a && ~b && c));
11. assign HEX0[5] = ~((~a && ~c && ~d) || (~a && b && ~c) || (~a && b && ~d) || (a && ~b && ~c) || (a && ~b && ~d) || (a && b && c));
12. assign HEX0[4] = ~((~b && ~d) || (~a && c && ~d) || (a && ~c && ~d));
13. assign HEX0[3] = ~((~b && ~d) || (a && ~c) || (~a && ~b && c) || (~a && c && ~d) || (b && ~c && d) || (a && b && d));
14. assign HEX0[2] = ~((d) || (~a && ~c) || (b && c) || (a && ~b));
15. assign HEX0[1] = ~((~b) || (~c && ~d) || (a &&~c) || (a &&~d) || (~a && c && d));
16. assign HEX0[0] = ~((~b && ~d) || (~a && c) || (b && d) || (a && ~c));
17.
18. assign HEX1[6] = ~(0);
19. assign HEX1[5] = ~((~a) || (~b && ~c));
20. assign HEX1[4] = ~((~a) || (~b && ~c));
21. assign HEX1[3] = ~((~a) || (~b && ~c));
22. assign HEX1[2] = ~(1);
23. assign HEX1[1] = ~(1);
24. assign HEX1[0] = ~((~a) || (~b && ~c));
25. endmodule
```

/*part6: Full Adder*/

```
1.  module lab1 (SW, LEDR);
2.  input [17:0] SW;
3.  output [17:0] LEDR;
```

```
4.  wire Cin, Cout;
5.  wire [3:0] A, B, S;
6.  wire [3:1] C;
7.
8.  assign {Cin, B, A} = SW [8:0];
9.  assign LEDR[4:0] = {Cout, S};
10.
11. assign S[0] = ((Cin)^(A[0] ^ B[0]));
12. assign C[1] = (~(A[0] ^ B[0]) & B[0]) | ((A[0] ^ B[0]) & Cin);
13. assign S[1] = ((C[1])^(A[1] ^ B[1]));
14. assign C[2] = (~(A[1] ^ B[1]) & B[1]) | ((A[1] ^ B[1]) & C[1]);
15. assign S[2] = ((C[2])^(A[2] ^ B[2]));
16. assign C[3] = (~(A[2] ^ B[2]) & B[2]) | ((A[2] ^ B[2]) & C[2]);
17. assign S[3] = ((C[3])^(A[3] ^ B[3]));
18. assign Cout = (~(A[3] ^ B[3]) & B[3]) | ((A[3] ^ B[3]) & C[3]);
19. endmodule
```