|  |
| --- |
| **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 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 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。

1. **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. wire Cin, Cout; 6. wire[3:0] A, B, S; 7. wire [3:1] C; 8. //Full Adder 9. assign A = SW [7:4]; 10. assign B = SW [3:0]; 11. assign Cin = SW[8]; 12. assign S[0] = ((Cin)^(A[0] ^ B[0])); 13. assign C[1] = (~(A[0] ^ B[0]) & B[0]) | ((A[0] ^ B[0]) & Cin); 14. assign S[1] = ((C[1])^(A[1] ^ B[1])); 15. assign C[2] = (~(A[1] ^ B[1]) & B[1]) | ((A[1] ^ B[1]) & C[1]); 16. assign S[2] = ((C[2])^(A[2] ^ B[2])); 17. assign C[3] = (~(A[2] ^ B[2]) & B[2]) | ((A[2] ^ B[2]) & C[2]); 18. assign S[3] = ((C[3])^(A[3] ^ B[3])); 19. assign Cout = (~(A[3] ^ B[3]) & B[3]) | ((A[3] ^ B[3]) & C[3]); 20. //BCD 21. //HEX1~0 22. wire a,b,c,d; 23. assign a = SW[3]; 24. assign b = SW[2]; 25. assign c = SW[1]; 26. assign d = SW[0]; 27. assign HEX0[6] = ~((b && ~c) || (b &&~d) || (a && ~c) || (a && b) || (~a && ~b && c)); 28. assign HEX0[5] = ~((~a && ~c && ~d) || (~a && b && ~c) || (~a && b && ~d) || (a && ~b && ~c) || (a && ~b && ~d) || (a && b && c)); 29. assign HEX0[4] = ~((~b && ~d) || (~a && c && ~d) || (a && ~c && ~d)); 30. assign HEX0[3] = ~((~b && ~d) || (a && ~c) || (~a && ~b && c) || (~a && c && ~d) || (b && ~c && d) || (a && b && d)); 31. assign HEX0[2] = ~((d) || (~a && ~c) || (b && c) || (a && ~b)); 32. assign HEX0[1] = ~((~b) || (~c && ~d) || (a &&~c) || (a &&~d) || (~a && c && d)); 33. assign HEX0[0] = ~((~b && ~d) || (~a && c) || (b && d) || (a && ~c)); 34. assign HEX1[6] = ~(0); 35. assign HEX1[5] = ~((~a) || (~b && ~c)); 36. assign HEX1[4] = ~((~a) || (~b && ~c)); 37. assign HEX1[3] = ~((~a) || (~b && ~c)); 38. assign HEX1[2] = ~(1); 39. assign HEX1[1] = ~(1); 40. assign HEX1[0] = ~((~a) || (~b && ~c)); 41. //HEX3~2 42. wire a2,b2,c2,d2; 43. assign a2 = SW[7]; 44. assign b2 = SW[6]; 45. assign c2 = SW[5]; 46. assign d2 = SW[4]; 47. assign HEX2[6] = ~((b2 && ~c2) || (b2 && ~d2) || (a2 && ~c2) || (a2 && b2) || (~a2 && ~b2 && c2)); 48. assign HEX2[5] = ~((~a2 && ~c2 && ~d2) || (~a2 && b2 && ~c2) || (~a2 && b2 && ~d2) || (a2 && ~b2 && ~c2) || (a2 && ~b2 && ~d2) || (a2 && b2 && c2)); 49. assign HEX2[4] = ~((~b2 && ~d2) || (~a2 && c2 && ~d2) || (a2 && ~c2 && ~d2)); 50. assign HEX2[3] = ~((~b2 && ~d2) || (a2 && ~c2) || (~a2 && ~b2 && c2) || (~a2 && c2 && ~d2) || (b2 && ~c2 && d2) || (a2 && b2 && d2)); 51. assign HEX2[2] = ~((d2) || (~a2 && ~c2) || (b2 && c2) || (a2 && ~b2)); 52. assign HEX2[1] = ~((~b2) || (~c2 && ~d2) || (a2 &&~c2) || (a2 &&~d2) || (~a2 && c2 && d2)); 53. assign HEX2[0] = ~((~b2 && ~d2) || (~a2 && c2) || (b2 && d2) || (a2 && ~c2)); 54. assign HEX3[6] = ~(0); 55. assign HEX3[5] = ~((~a2) || (~b2 && ~c2)); 56. assign HEX3[4] = ~((~a2) || (~b2 && ~c2)); 57. assign HEX3[3] = ~((~a2) || (~b2 && ~c2)); 58. assign HEX3[2] = ~(1); 59. assign HEX3[1] = ~(1); 60. assign HEX3[0] = ~((~a2) || (~b2 && ~c2)); 61. //HEX5~4 5bit-BCD 62. wire a3,b3,c3,d3; 63. assign a3 = Cout; 64. assign b3 = S[3]; 65. assign c3 = S[2]; 66. assign d3 = S[1]; 67. assign e3 = S[0]; 68. assign HEX4[6] = ~((b3 & ~d3) | (~b3 & ~c3 & d3) | (~b3 & d3 & ~e3) | (~a3 & c3 & ~d3) | (~a3 & b3 & c3) | (a3 & ~c3 & ~e3) | (a3 & ~b3 & d3)); 69. 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)); 70. assign HEX4[4] = ~((~b3 & ~c3 & ~e3) | (~b3 & d3 & ~e3) | (~c3 & d3 & ~e3) | (a3 & c3 & ~e3) | (~a3 & b3 & ~d3 & ~e3)); 71. 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)); 72. assign HEX4[2] = ~(e3 | (~c3 & ~d3) | (b3 & d3) | (a3 & ~c3) | (a3 & ~d3) | (~a3 & ~b3 & c3)); 73. 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)); 74. 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)); 75. assign HEX5[6] = ~((a3 & c3) | (a3 & b3)); 76. assign HEX5[5] = ~((~a3 & ~b3) | (~a3 & ~c3 & ~d3)); 77. assign HEX5[4] = ~((~a3 & ~b3) | (~b3 & c3) | (~a3 & ~c3 & ~d3) | (a3 & c3 & ~d3) | (a3 & b3 & ~c3)); 78. assign HEX5[3] = ~((~a3 & ~b3) | (~b3 & c3) | (a3 & b3) | (~a3 & ~c3 & ~d3)); 79. assign HEX5[2] = ~(~a3 | (~b3 & ~c3) | (b3 & c3 & d3)); 80. assign HEX5[1] = ~(1); 81. assign HEX5[0] = ~((~a3 & ~b3) | (~b3 & c3) | (a3 & b3) | (~a3 & ~c3 & ~d3)); 82. //ERROR LEDR[9] 83. assign LEDR[9] = ((a & c) | (a & b)) | ((a2 & c2) | (a2 & b2)); 84. endmodule |

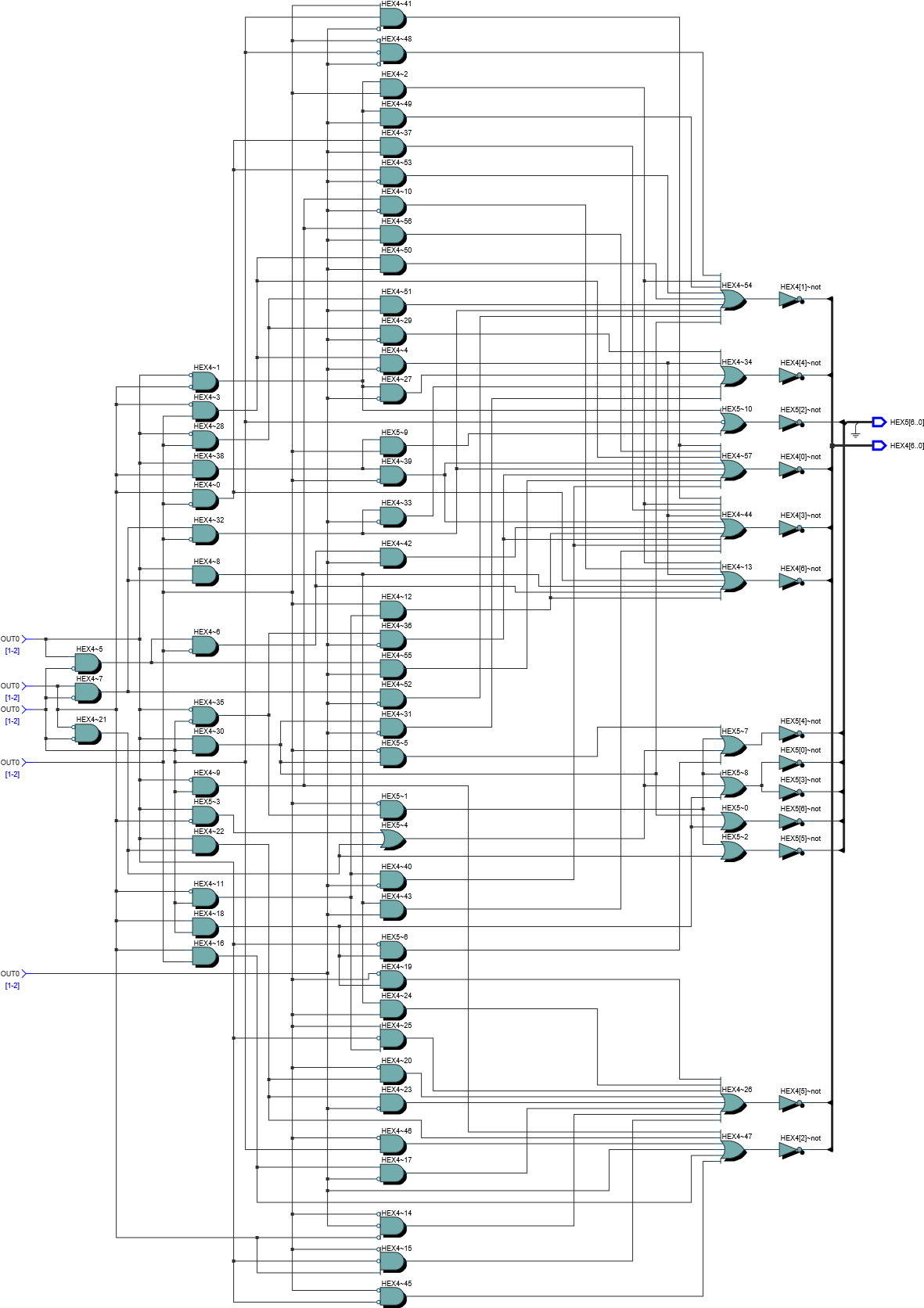
1. **實驗結果照片(optional)**

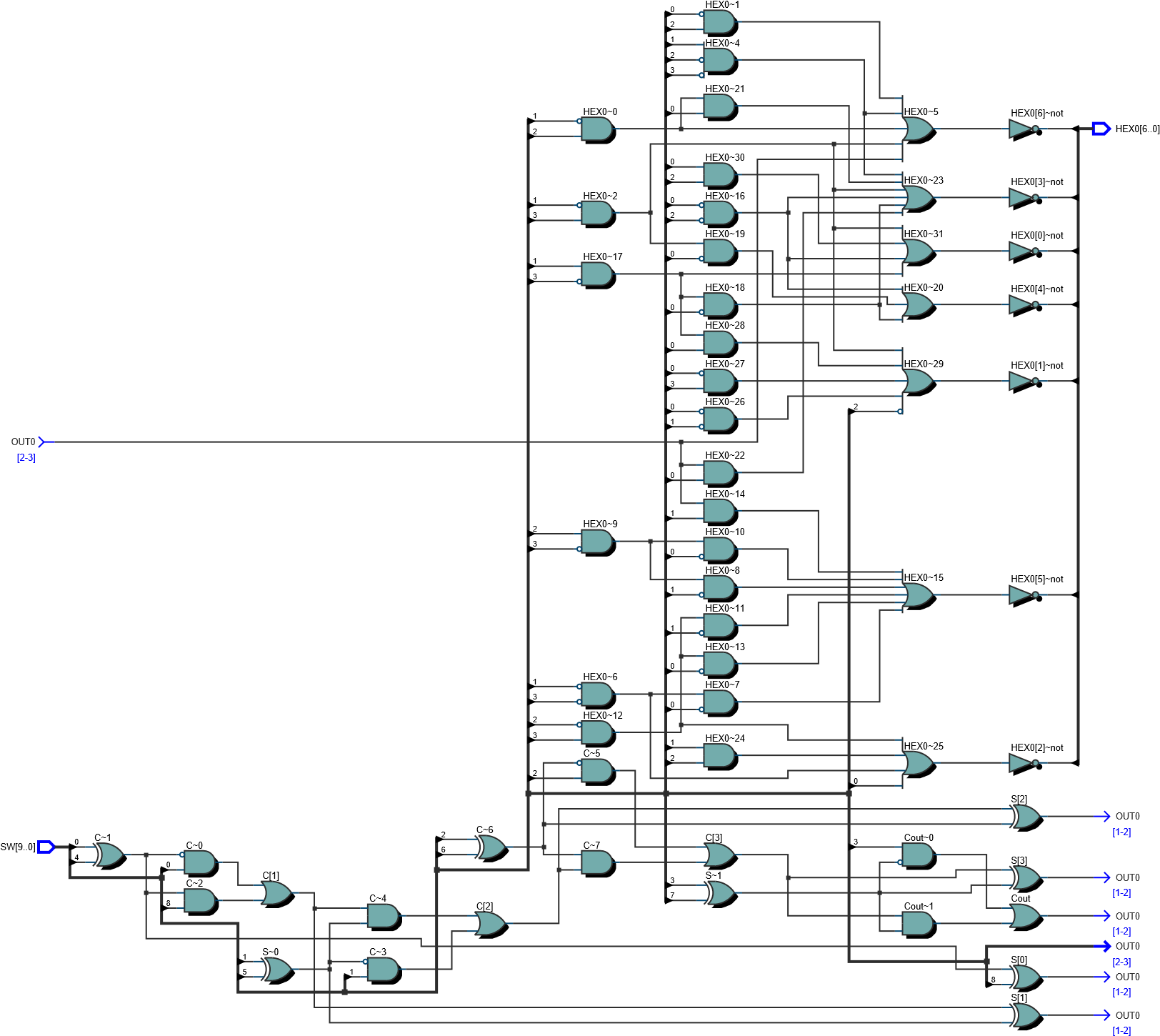
實驗 part7 實作Add two 1-digit numbers BCD

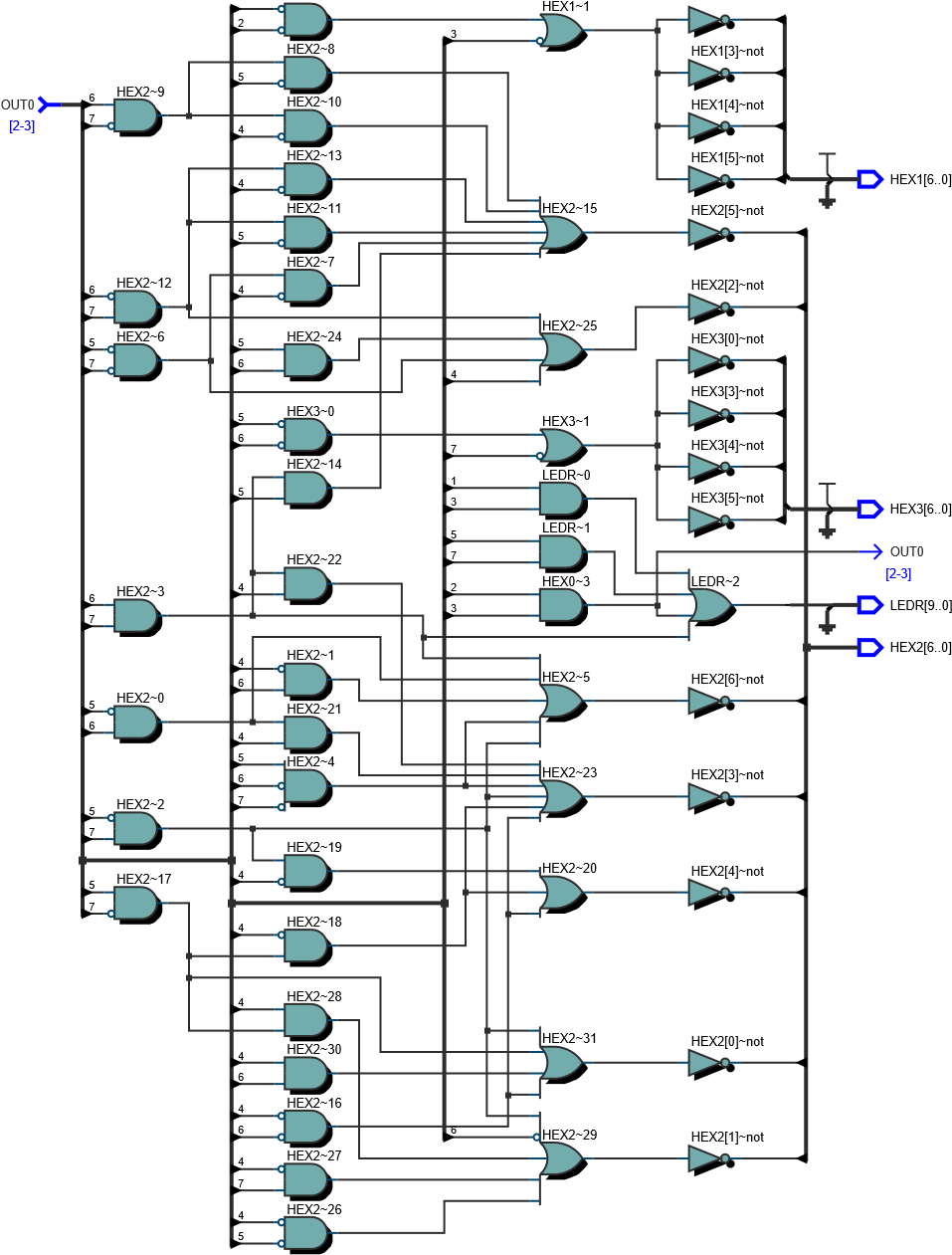
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

1. **RTL布局(optional)**

Tools >> Netlist viewer >> RTL viewer.







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

|  |
| --- |
| /\*part5: BCD binary coded decimal\*/ |
| 1. module lab1 (SW,HEX1,HEX0); 2. input [3:0] SW; 3. output [6:0] HEX1, HEX0; 4. assign a = SW[3]; 5. assign b = SW[2]; 6. assign c = SW[1]; 7. assign d = SW[0]; 8. //HEX is common anode configuration 9. assign HEX0[6] = ~((b && ~c) || (b &&~d) || (a && ~c) || (a && b) || (~a && ~b && c)); 10. assign HEX0[5] = ~((~a && ~c && ~d) || (~a && b && ~c) || (~a && b && ~d) || (a && ~b && ~c) || (a && ~b && ~d) || (a && b && c)); 11. assign HEX0[4] = ~((~b && ~d) || (~a && c && ~d) || (a && ~c && ~d)); 12. assign HEX0[3] = ~((~b && ~d) || (a && ~c) || (~a && ~b && c) || (~a && c && ~d) || (b && ~c && d) || (a && b && d)); 13. assign HEX0[2] = ~((d) || (~a && ~c) || (b && c) || (a && ~b)); 14. assign HEX0[1] = ~((~b) || (~c && ~d) || (a &&~c) || (a &&~d) || (~a && c && d)); 15. assign HEX0[0] = ~((~b && ~d) || (~a && c) || (b && d) || (a && ~c)); 16. assign HEX1[6] = ~(0); 17. assign HEX1[5] = ~((~a) || (~b && ~c)); 18. assign HEX1[4] = ~((~a) || (~b && ~c)); 19. assign HEX1[3] = ~((~a) || (~b && ~c)); 20. assign HEX1[2] = ~(1); 21. assign HEX1[1] = ~(1); 22. assign HEX1[0] = ~((~a) || (~b && ~c)); 23. 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. assign {Cin, B, A} = SW [8:0]; 8. assign LEDR[4:0] = {Cout, S}; 9. assign S[0] = ((Cin)^(A[0] ^ B[0])); 10. assign C[1] = (~(A[0] ^ B[0]) & B[0]) | ((A[0] ^ B[0]) & Cin); 11. assign S[1] = ((C[1])^(A[1] ^ B[1])); 12. assign C[2] = (~(A[1] ^ B[1]) & B[1]) | ((A[1] ^ B[1]) & C[1]); 13. assign S[2] = ((C[2])^(A[2] ^ B[2])); 14. assign C[3] = (~(A[2] ^ B[2]) & B[2]) | ((A[2] ^ B[2]) & C[2]); 15. assign S[3] = ((C[3])^(A[3] ^ B[3])); 16. assign Cout = (~(A[3] ^ B[3]) & B[3]) | ((A[3] ^ B[3]) & C[3]); 17. endmodule |