今天的實驗是有關加法器

第一個實驗:實作 8bit 加法器

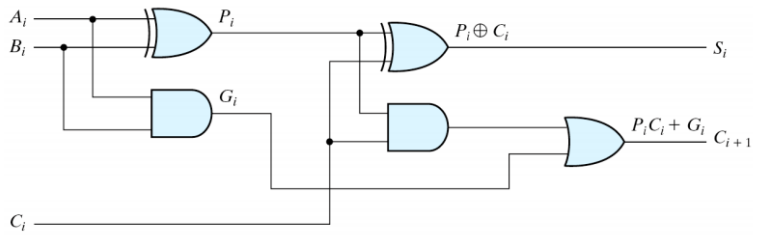
Carry Computation

$$A_{i+3}\,A_{i+2}\,A_{i+1}\,A_i + B_{i+3}\,B_{i+2}\,B_{i+1}\,B_i \Rightarrow C_{i+4}\,C_{i+3}\,C_{i+2}\,C_{i+1}$$

Let $G_i = A_i\,B_i$, $P_i = A_i \oplus B_i$

$$C_{i+1} = A_i\,B_i + C_i(A_i \oplus B_i) = G_i + P_i\,C_i,$$
$$C_{i+2} = G_{i+1} + P_{i+1}\,C_{i+1}, \quad C_{i+3} = G_{i+2} + P_{i+2}\,C_{i+2}, \quad C_{i+4} = G_{i+3} + P_{i+3}\,C_{i+3}$$

$$C_{i+1} = G_i + P_i\,C_i$$
$$C_{i+2} = G_{i+1} + P_{i+1}\,G_i + P_{i+1}\,P_i\,C_i$$
$$C_{i+3} = G_{i+2} + P_{i+2}\,G_{i+1} + P_{i+2}\,P_{i+1}\,G_i + P_{i+2}\,P_{i+1}\,P_i\,C_i$$
$$C_{i+4} = G_{i+3} + P_{i+3}\,G_{i+2} + P_{i+3}\,P_{i+2}\,G_{i+1} + P_{i+3}\,P_{i+2}\,P_{i+1}\,G_i + P_{i+3}\,P_{i+2}\,P_{i+1}\,P_i\,C_i$$



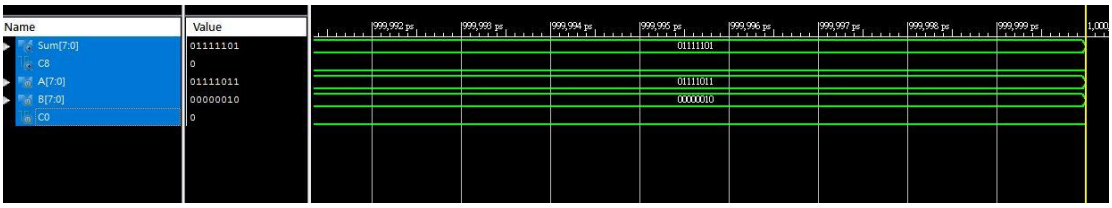
將 carry 實現

```
22  module mod_4(output [3:0]Sum , output C4 , input [3:0]A,input [3:0] B ,input C0);
23     wire C1,C2,C3;
24     wire P0,G0,P0C0;
25     xor(P0,A[0],B[0]);
26     and(G0,A[0],B[0]);
27     and(P0C0,C0,P0);
28     or(C1,P0C0,G0);
29
30     wire G1,P1,P1G0,P1P0C0;
31     and(G1,A[1],B[1]);
32     xor(P1,A[1],B[1]);
33     and(P1G0,P1,G0);
34     and(P1P0C0,P1,P0,C0);
35     or(C2,G1,P1G0,P1P0C0);
36
37     wire G2,P2,P2G1,P2P1G0,P2P1P0C0;
38     and(G2,A[2],B[2]);
39     xor(P2,A[2],B[2]);
40     and(P2G1,P2,G1);
41     and(P2P1G0,P2,P1,G0);
42     and(P2P1P0C0,P2,P1,P0,C0);
43     or(C3,G2,P2G1,P2P1G0,P2P1P0C0);
44
45     wire G3,P3,P3G2,P3P2G1,P3P2P1G0,P3P2P1P0C0;
46     and(G3,A[3],B[3]);
47     xor(P3,A[3],B[3]);
48     and(P3G2,P3,G2);
49     and(P3P2G1,P3,P2,G1);
50     and(P3P2P1G0,P3,P2,P1,G0);
51     and(P3P2P1P0C0,P3,P2,P1,P0,C0);
52     or(C4,G3,P3G2,P3P2G1,P3P2P1G0,P3P2P1P0C0);
53
54     xor(Sum[0],P0,C0);
55     xor(Sum[1],P1,C1);
56     xor(Sum[2],P2,C2);
57     xor(Sum[3],P3,C3);
58
59  endmodule
```

此為 4bit 的加法器

因此需要合併兩個 4bit 加法器

```
62  module mod_8(output [7:0]Sum , output C8 , input [7:0]A,input [7:0]B ,input C0);
63     wire C4;
64     mod_4 low(Sum[3:0],C4,A[3:0],B[3:0],C0);
65     mod_4 high(Sum[7:4],C8,A[7:4],B[7:4],C4);
66
67  endmodule
68
```



由此實現 8bit adder with Carry Lookahead

第二個實驗:實現 BCD 加法器

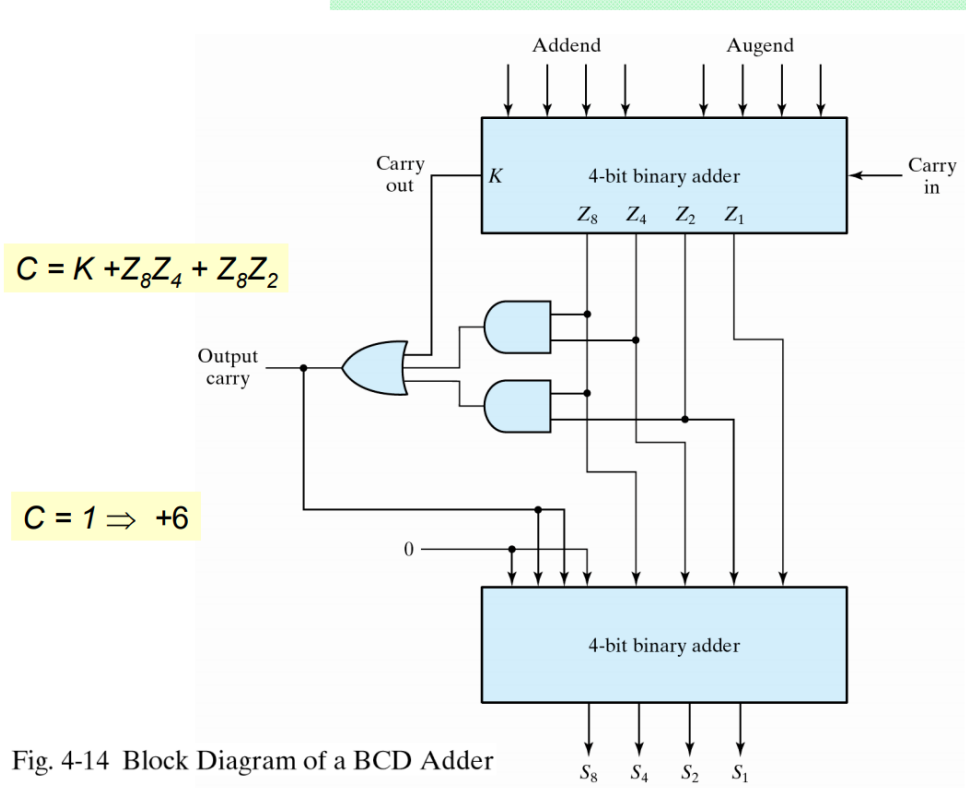
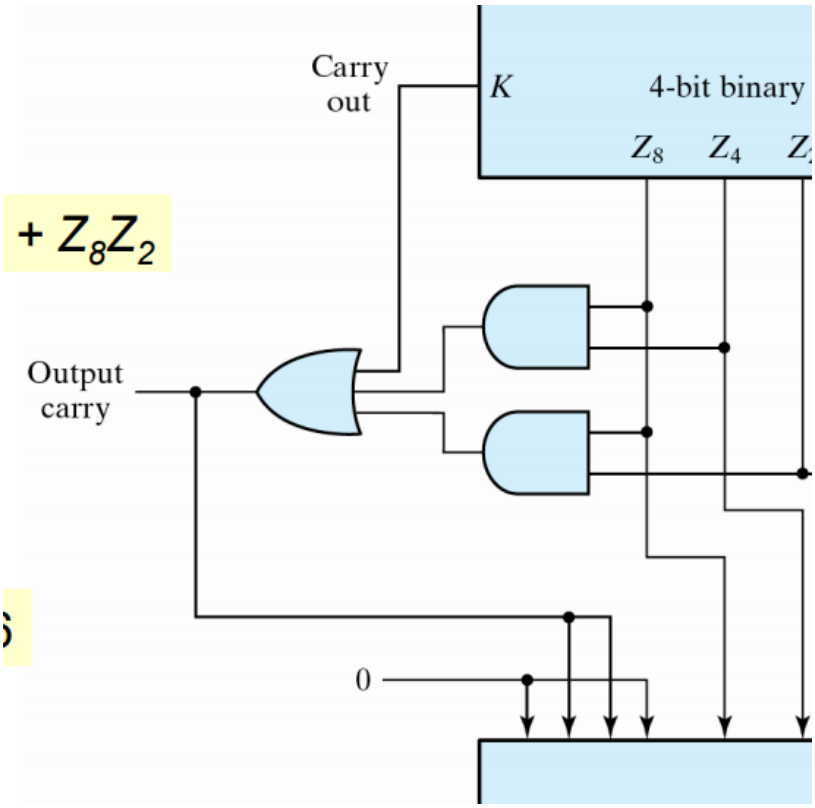


Fig. 4-14 Block Diagram of a BCD Adder

因為一個 BCD 碼是 4bit，因此可以使用前一個實驗的加法器

重點在判斷這個數字是否大於 9



我這次使用的是 DataFlow_Modeling

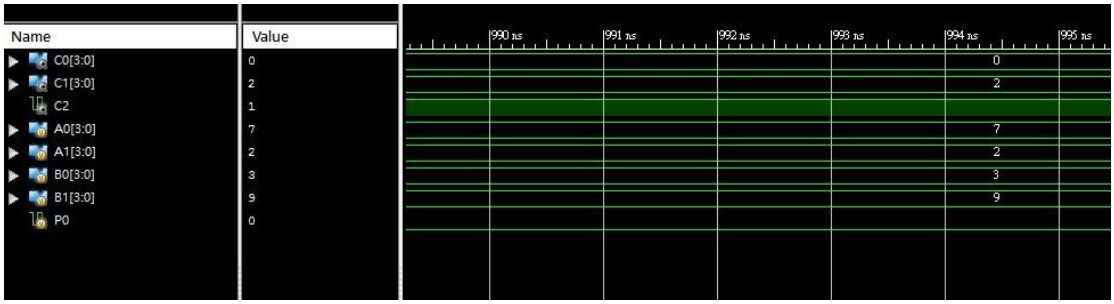
```
59
60  module mod(input [3:0]A0,input [3:0]A1,input [3:0]B0 ,input [3:0]B1,output [3:0]C0,output [3:0]C1,output C2,input P0);
61     wire add0;
62     wire add1;
63     wire add2;
64     wire add3;
65     wire CA0;
66     wire ca0,ca1,or1;
67     wire ca2,ca3,or2;
68     wire [3:0]CX;
69     wire [3:0]CY;
70     wire [3:0]Low_6;
71     wire [3:0]High_6;
72     mod_4 low(CX,add0,A0,B0,P0);
73     assign ca0 = CX[3]&CX[1];
74     assign ca1 = CX[3]&CX[2];
75     assign or1 = add0|ca0|ca1;
76     assign Low_6 = {1'b0,or1,or1,1'b0};
77     mod_4 low_x(C0,add1,CX,Low_6,P0);
78     assign CA0=add0|add1;
79
80     mod_4 high(CY,add2,A1,B1,CA0);
81     assign ca2 = CY[3]&CY[1];
82     assign ca3 = CY[3]&CY[2];
83     assign or2 = add2|ca2|ca3;
84     assign High_6 = {1'b0,or2,or2,1'b0};
85     mod_4 high_x(C1,add3,CY,High_6,P0);
86     assign C2 = add2|add3;
87
88
89  endmodule
90
```

所以使用 assign 語法

判斷大於 9 時 assign 數字 6(line 76.84)

小於 10 時 assign 數字 0

再將數字跟剛剛 assign 數字相加，得到該位的數字



實作 BCD 加法器

學到幾個新知識:

1. wire 可以陣列宣告
2. reg 只能在 Behavior Modeling 中使用
3. dataflow 的 assign 可以
    - assign 的值也可以用大括弧來串聯 bit。
        - `assign a = { 1'b0, 1'b1 }; // a = 2'b01`
        - `assign a = { 2{2'b10} }; // a = 4'b1010`

這樣指定數字..

參考來源:

1. https://hackmd.io/@dppa1008/BJWS5_B_G?type=view