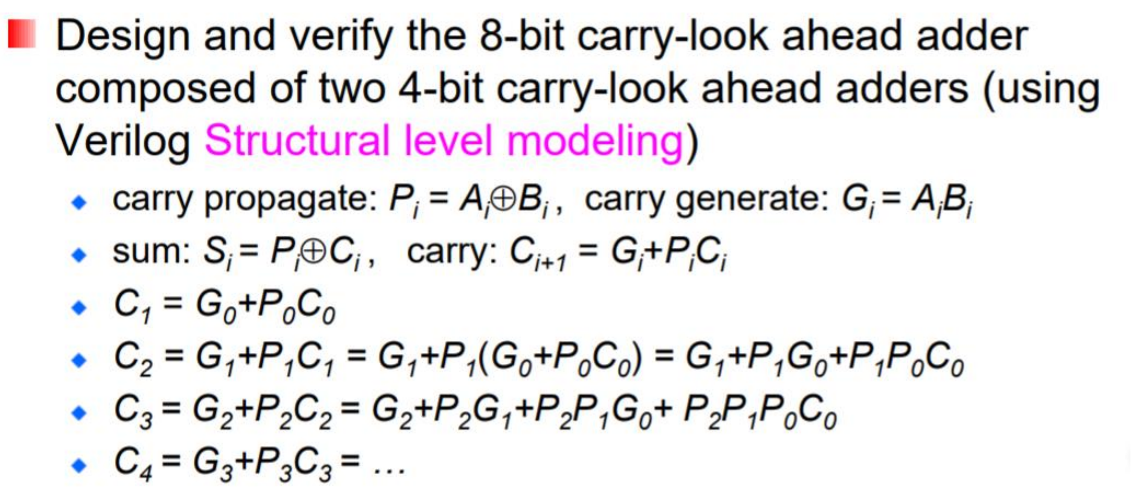
實驗主題(Lab08)

B083022053

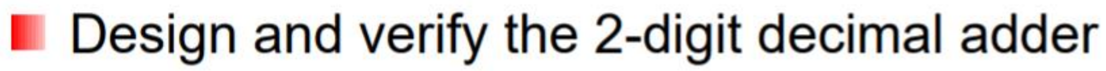
黃啟桓

Exercise 1:



- Design and verify the 8-bit carry-look ahead adder composed of two 4-bit carry-look ahead adders (using Verilog Structural level modeling)
  - carry propagate: $P_i = A_i \oplus B_i$, carry generate: $G_i = A_iB_i$
  - sum: $S_i = P_i \oplus C_i$, carry: $C_{i+1} = G_i + P_iC_i$
  - $C_1 = G_0 + P_0C_0$
  - $C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$
  - $C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
  - $C_4 = G_3 + P_3C_3 = \ldots$

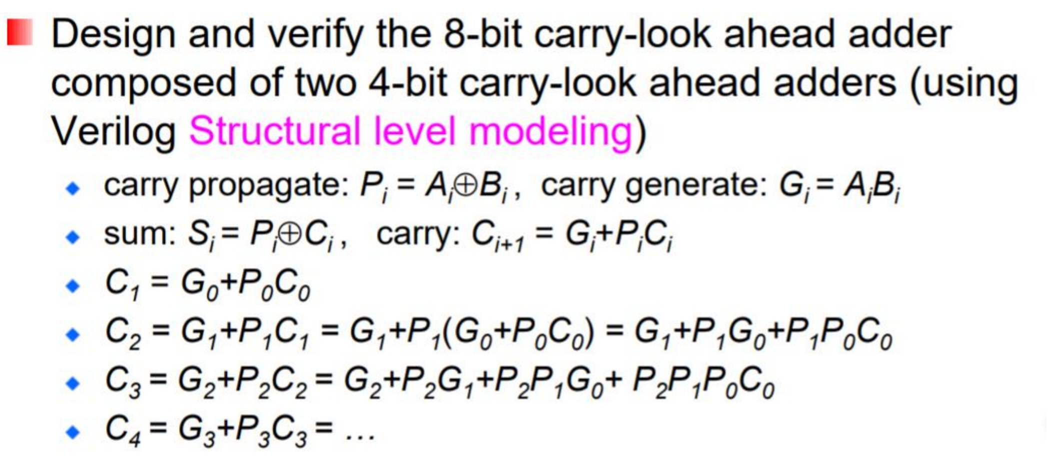
Exercise 2:



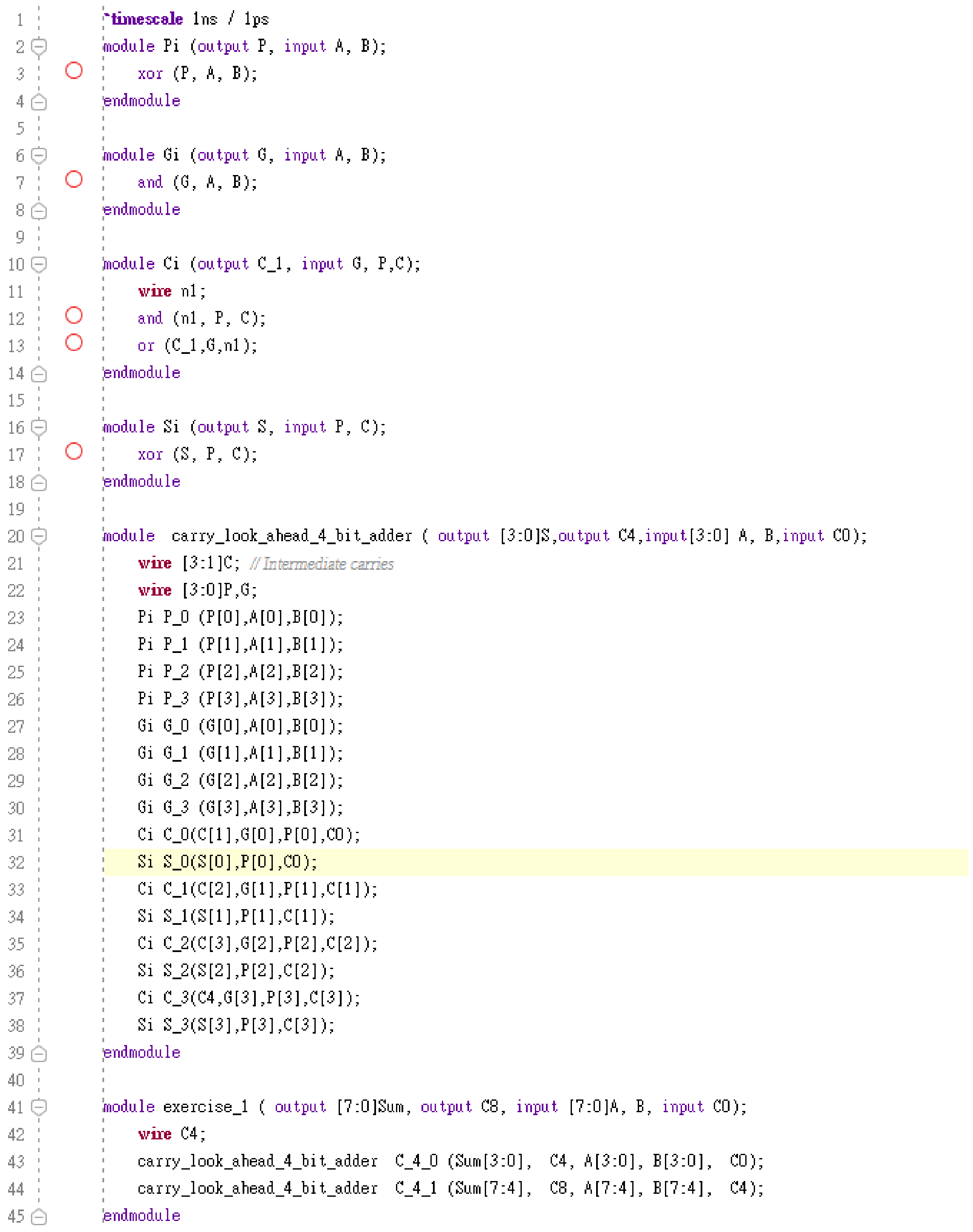
- Design and verify the 2-digit decimal adder

實驗內容

一、Exercise 1:

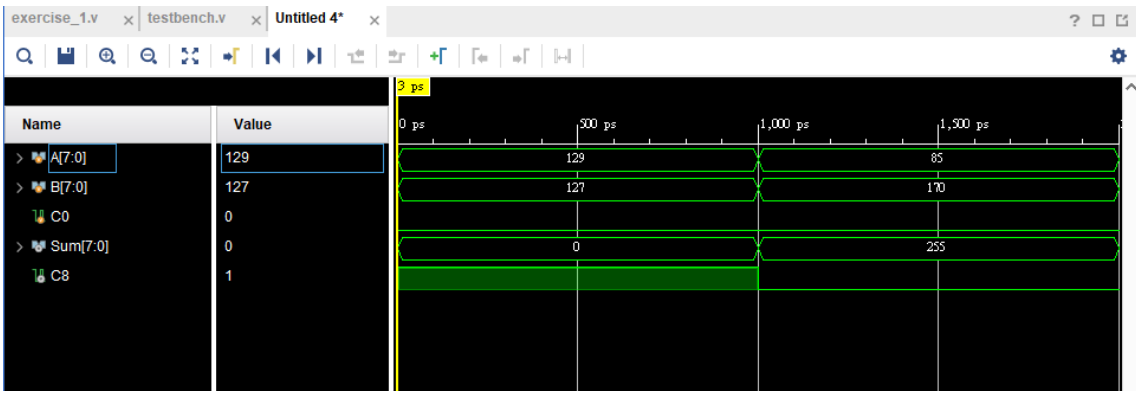


- Design and verify the 8-bit carry-look ahead adder composed of two 4-bit carry-look ahead adders (using Verilog Structural level modeling)
  - carry propagate: $P_i = A_i \oplus B_i$, carry generate: $G_i = A_iB_i$
  - sum: $S_i = P_i \oplus C_i$, carry: $C_{i+1} = G_i + P_iC_i$
  - $C_1 = G_0 + P_0C_0$
  - $C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$
  - $C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
  - $C_4 = G_3 + P_3C_3 = \ldots$



```verilog
`timescale 1ns / 1ps
module Pi (output P, input A, B);
    xor (P, A, B);
endmodule

module Gi (output G, input A, B);
    and (G, A, B);
endmodule

module Ci (output C_1, input G, P,C);
    wire n1;
    and (n1, P, C);
    or (C_1,G,n1);
endmodule

module Si (output S, input P, C);
    xor (S, P, C);
endmodule

module  carry_look_ahead_4_bit_adder ( output [3:0]S,output C4,input[3:0] A, B,input C0);
    wire [3:1]C; // Intermediate carries
    wire [3:0]P,G;
    Pi P_0 (P[0],A[0],B[0]);
    Pi P_1 (P[1],A[1],B[1]);
    Pi P_2 (P[2],A[2],B[2]);
    Pi P_3 (P[3],A[3],B[3]);
    Gi G_0 (G[0],A[0],B[0]);
    Gi G_1 (G[1],A[1],B[1]);
    Gi G_2 (G[2],A[2],B[2]);
    Gi G_3 (G[3],A[3],B[3]);
    Ci C_0(C[1],G[0],P[0],C0);
    Si S_0(S[0],P[0],C0);
    Ci C_1(C[2],G[1],P[1],C[1]);
    Si S_1(S[1],P[1],C[1]);
    Ci C_2(C[3],G[2],P[2],C[2]);
    Si S_2(S[2],P[2],C[2]);
    Ci C_3(C4,G[3],P[3],C[3]);
    Si S_3(S[3],P[3],C[3]);
endmodule

module exercise_1 ( output [7:0]Sum, output C8, input [7:0]A, B, input C0);
    wire C4;
    carry_look_ahead_4_bit_adder  C_4_0 (Sum[3:0],  C4, A[3:0], B[3:0],  C0);
    carry_look_ahead_4_bit_adder  C_4_1 (Sum[7:4],  C8, A[7:4], B[7:4],  C4);
endmodule
```

carry-look ahead adder 延遲比上禮拜的 full adder 少



Name	Value	0–1,000 ps	1,000–2,000 ps
A[7:0]	129	129	85
B[7:0]	127	127	170
C0	0		
Sum[7:0]	0	0	255
C8	1		

127+129=256 =2^8

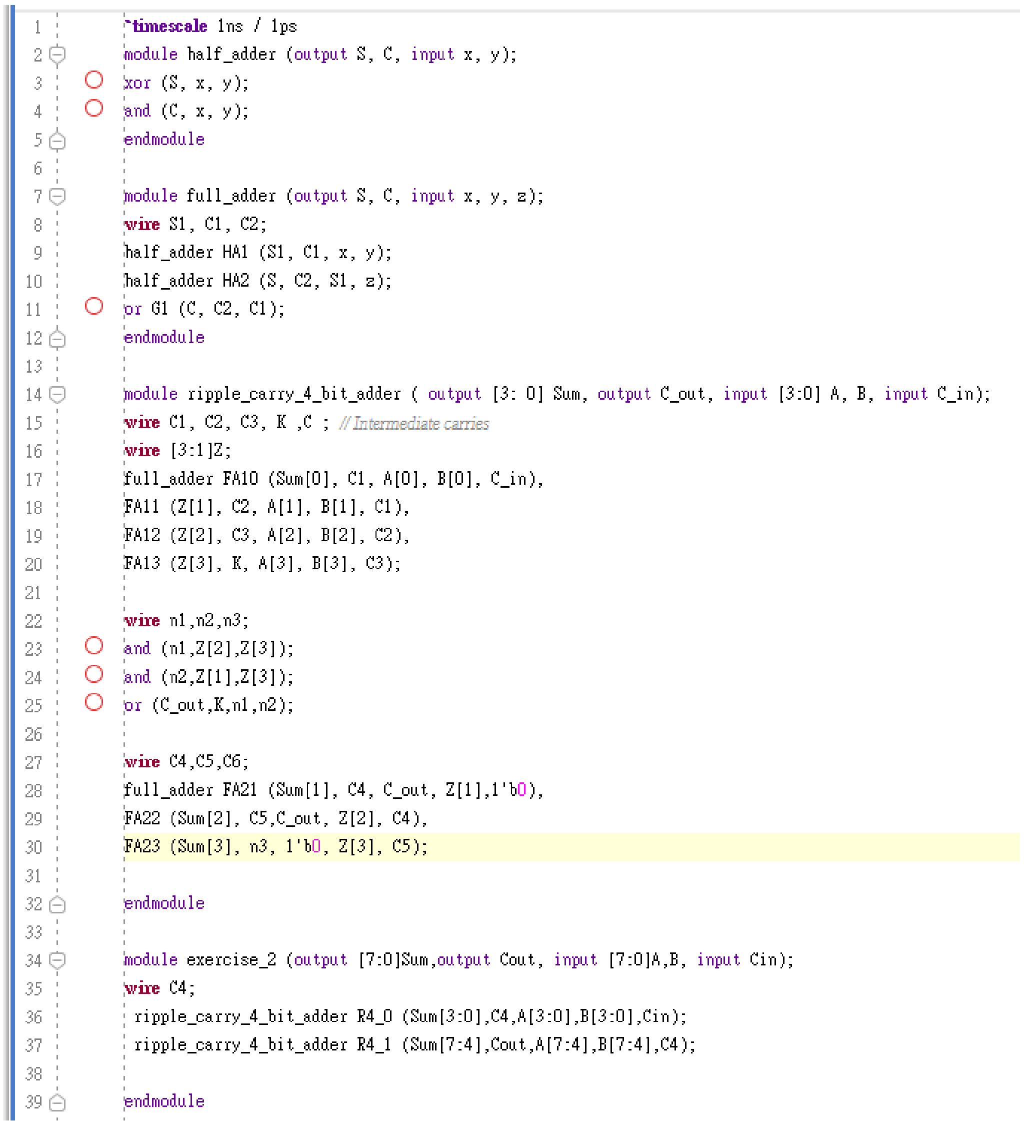
85+170=255 <2^8

二、Exercise　2:

- Design and verify the 2-digit decimal adder

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

$$C = 1 \Rightarrow \ +6$$



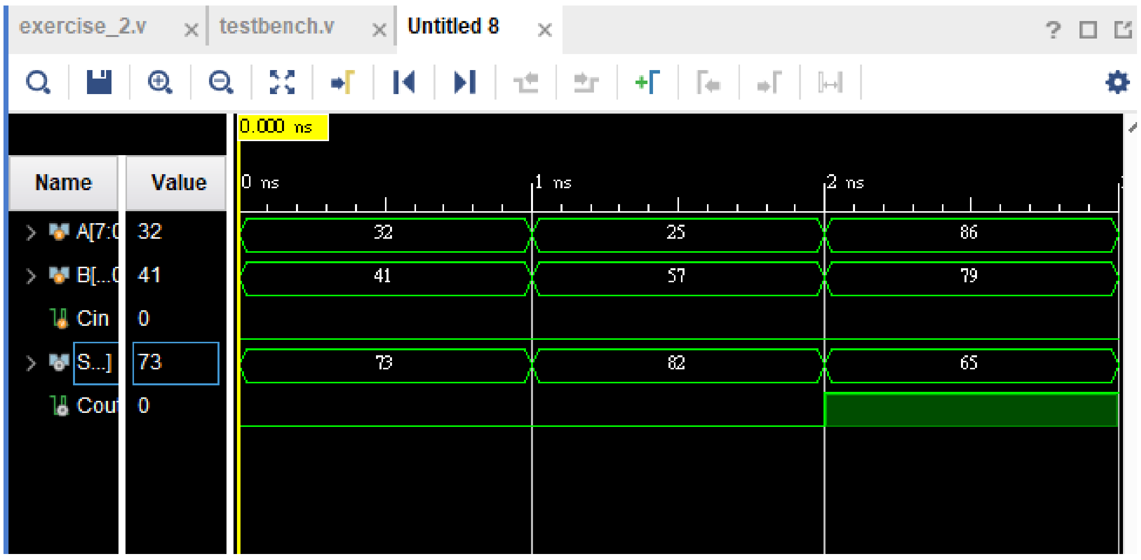
```verilog
`timescale 1ns / 1ps
module half_adder (output S, C, input x, y);
xor (S, x, y);
and (C, x, y);
endmodule

module full_adder (output S, C, input x, y, z);
wire S1, C1, C2;
half_adder HA1 (S1, C1, x, y);
half_adder HA2 (S, C2, S1, z);
or G1 (C, C2, C1);
endmodule

module ripple_carry_4_bit_adder ( output [3: 0] Sum, output C_out, input [3:0] A, B, input C_in);
wire C1, C2, C3, K ,C ; // Intermediate carries
wire [3:1]Z;
full_adder FA10 (Sum[0], C1, A[0], B[0], C_in),
FA11 (Z[1], C2, A[1], B[1], C1),
FA12 (Z[2], C3, A[2], B[2], C2),
FA13 (Z[3], K, A[3], B[3], C3);

wire n1,n2,n3;
and (n1,Z[2],Z[3]);
and (n2,Z[1],Z[3]);
or (C_out,K,n1,n2);

wire C4,C5,C6;
full_adder FA21 (Sum[1], C4, C_out, Z[1],1'b0),
FA22 (Sum[2], C5,C_out, Z[2], C4),
FA23 (Sum[3], n3, 1'b0, Z[3], C5);

endmodule

module exercise_2 (output [7:0]Sum,output Cout, input [7:0]A,B, input Cin);
wire C4;
 ripple_carry_4_bit_adder R4_0 (Sum[3:0],C4,A[3:0],B[3:0],Cin);
 ripple_carry_4_bit_adder R4_1 (Sum[7:4],Cout,A[7:4],B[7:4],C4);

endmodule
```

10 進位加法器進位需要+6，也就是+4+2，第 2、3、4bit 再執行一次加法

32 + 41 = 073

25 + 57 = 082

86 + 79 = 165



Name	Value	0–1 ns	1–2 ns	2–3 ns
A[7:0	32	32	25	86
B[...0	41	41	57	79
Cin	0			
S...]	73	73	82	65
Cout	0			

實驗心得

1. Structural level modeling 中邏輯閘使用函數表示
2. Structural level modeling 可以很直觀的轉換成 Schematic
3. Structural level modeling 需要宣告邏輯閘之間的線路
4. 所有的 modeling 都需要注意該邏輯閘是否有交換律或結合律