Design of CLA

我用 P[3:0]，G[3:0] 分別代表 Propagate function 和 Generate function，最後根據 C_i = G_(i-1) + P_(i-1)C_(i-1) 推出以下式子:

P = A ^ B

G = A & B

C[0] = Cin

C[1] = G[0]|(P[0]&C[0])

C[2] = G[1]|(P[1]&G[0])|(P[1]&P[0]&C[0])

C[3] = G[2]|(P[2]&G[1])|(P[2]&P[1]&G[0])|(P[2]&P[1]&P[0]&C[0])

Cout = G[3]|(P[3]&G[2])|(P[3]&P[2]&G[1])|(P[3]&P[2]&P[1]&G[0])|(P[3]&P[2]&P[1]&P[0]&C[0])

S = P ^ G

16bit Adder

我的想法是利用模組例化(Module Instantiation)的方式呼叫 4 次 CLA module:

```
Wire [2:0] C;
CLA_4bit a(A[3:0], B[3:0], Cin, S[3:0], C[0]);
CLA_4bit b(A[7:4], B[7:4], C[0], S[7:4], C[1]);
CLA_4bit c(A[11:8], B[11:8], C[1], S[11:8], C[2]);
CLA_4bit d(A[15:12], B[15:12], C[2], S[15:12], Cout);
```

Implementation of ALU

對於

Mode = 0，Y = A << 1;

Mode = 1，Y = A <<< 1;

Mode = 2，Y = A >> 1;

Mode = 3，Y = {A[15], A[15:1] }; 利用{}串接是因為我當時使用>>>但是答案是錯的

Mode = 4，我先在 always 外面呼叫 16bit_Adder，用事先新建的 wire 變數去連接答案，如果跑到這個 case 就直接賦值給 Y，Cout，Overflow

Mode = 5，同 mode=4，只是我把 B 改成-B

Mode = 6，Y = A&B;

Mode = 7，Y = A|B;
Mode = 8，Y = ~A;
Mode = 9，Y = A ^ B;
Mode = 10，Y = ~(A ^ B);
Mode = 11，Y = ~(A | B);
Mode = 12，Y = 1<<A[3:0]; (利用左移運算實作 decoder)
Mode = 13，因為是 signed int 數字比大小，所以先分開討論最高位的情況，剩下的就直接用<來算答案

```
if(A[15]==0 && B[15]==1) begin
        Y = 0;
end
else if(A[15]==1 && B[15]==0) begin
        Y = 1;
end
else begin
        Y = (A<B);
End
```

Mode = 14，Y = B;
Mode = 15，實作 priority encoder，我是利用 casex，直接寫了 16 種情況去判斷答案

The problems you faced and how you deal with it.

在實作 ALU 的時候我遇到的最大問題有 2 個，第一個是 Module Instantiation 的部分，我原本想在 always block 裡面呼叫 16bit_adder，編譯的時候出現 error，上網查才發現不能這麼做，所以才想到要在 always block 外面先把答案存起來，到 always block 裡面的時候就可以直接賦值。

第二個是在實作 mode = 15 的時候，一開始我的想法是在裡面跑一個 for loop 去算，但在 testbench 的時候發現答案一直是錯的，後來我想說改成 case，發現答案一樣錯誤，上網查了一下對照自己的程式碼才知道如果有用到 x 的話要使用 casex。

The questions you want to ask TAs.

希望 Hackmd 有更詳細一點的 verilog 語法或是觀念講解(?