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(?