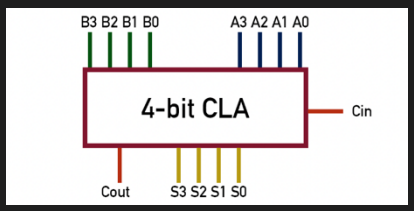
Mary Madeline Nicole 109006205

Logic Design Lab 2 ALU and Carry-Lookahead Adder - Report

1. *Design of 4-bit carry lookahead adder*

Carry Lookahead Adder is a fast adder to add two n-bit binary numbers with a generate function (Gi = Ai&Bi) and Propagate function (Pi = Ai^Bi). With this function, I compute the carry bits such that ($C_{i+1} = G_i + P_iC_i$), where all the carries from C_1 until C_4 can be computed in parallel form A,B and the input carry C_0,after a total of three gate delays. In the module CLA_4bit, I assign G = A&B and P = A^B. Then, I assigned Cin as C[0], and assigned the rest accordingly to the Carry bits function. Whereas the Cout would C_4 in the form of C_0. Then, I assigned S (the sum) into P^C.

```
module CLA_4bit(A, B, Cin, S, Cout);

    parameter n = 4;
    input [n - 1: 0] A, B;
    input Cin;

    output [n - 1: 0] S;
    output Cout;

    wire [3:0] G,P,C;

    //Generate function
    assign G = A & B; // generate
    //Propagate
    assign P = A ^ B; // propagate
    //Computation of carry bits
    assign C[0] = Cin;
    assign C[1] = G[0] | (P[0]&C[0]);
    assign C[2] = G[1] | (P[1]&G[0]) | (P[1]&P[0]&C[0]);
    assign C[3] = G[2] | (P[2]&G[1]) | (P[2]&P[1]&G[0]) | (P[2]&P[1]&P[0]&C[0]);
    //Compute Cous and S
    assign 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]);
    assign S = P ^ C;

endmodule
```



2. *Constructing 16-bit adder with 4-bit CLAs*

I constructed a 16-bit adder with 4 4-bit CLAs by designing a propagate function and a generate function for each group. Where I assign the S to be the Sum for every 4 bits. Every group has 4 bits, which are [3:0], [7:4], [11:8], [15:12]. The first group propagate function $P_{0\text{-}3} = P_0P_1P_2P_3$ and the generate function is $G_{0\text{-}3} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2G_1G_0$. The group propagate and generate functions for the rest of the other groups can be defined similarly. As I already made a 4 bit CLA, I used the module to

make a 16 bit adder directly by implementing the module. After finding the sum of every 4 bit, I then assigned the sum accordingly from the 4 CLAs from the most significant to the least significant bit.

```
module Adder_16bit(A, B, Cin, S, Cout);

    parameter n = 16;
    parameter m = 4;

    input [n - 1: 0] A, B;
    input Cin;

    output [n - 1: 0] S;
    output Cout;

    //Please refer to P32 of slides.
    wire C4, C8, C12;
    wire [m - 1: 0] S0_3, S4_7, S8_11, S12_15;
    assign S = {S12_15, S8_11, S4_7, S0_3};

    CLA_4bit cla0(A[3:0], B[3:0], Cin, S0_3, C4);
    CLA_4bit cla1(A[7:4], B[7:4], C4, S4_7, C8);
    CLA_4bit cla2(A[11:8], B[11:8], C8, S8_11, C12);
    CLA_4bit cla3(A[15:12], B[15:12], C12, S12_15, Cout);

endmodule
```

3. *Design and implementation of ALU*

- Logical shift A by 1-bit to the left and to the right

  I assign Y into A << 1'b1 to shift left by 1-bit, and assign A>>1'b1 to shift right by 1-bit. I also assigned the Cout and Overflow to 0.

- Arithmetic shift left by 1-bit

  Arithmetic shift, I assign A<<<1'b1 into Y to shift A left by 1 bit, then assigned the Cout and Overflow to 0. But for the arithmetic shift right by 1-bit, I assign A>>>1'b1 to Y and make Y[15] to 1 to maintain the sign. Then assigned the Cout and Overflow to 0.

- Add B to A with Adder 16 bit

  I used the 16 bit Adder to add B to A. But first, I wired 16 bit add_AB to store the sum of A + B, and a 1 bit wire of CoutAdd to store the carry bit after the operator. Inside the Adder16bit, I assigned the parameter as (A,B,Cin,add_AB,CoutAdd). I then assigned the Cout into CoutAdd, because it is the Carry bit after adding B to A. For the Overflow, I made a K-map of A,B,Sum

and Overflow for addition and found the Boolean expression for overflow, which is (!A!BY) + (AB!Y).

- Subtract B from A with adder 16 bit

  I used the 16 bit Adder to substract B from A. But first, I wired 16 bit sub_AB to store the sum of A + (-B), and a 1 bit wire of CoutSub to store the carry bit after the operator. Because adding a negative would mean that we need to find the 2's complement of the binary digit, instead of putting (A,B,Cin,Sum,Cout) inside the Adder16bit, I changed the B into complement directly at the parameter, which became (A,(~B)+1,Cin,Sub_AB,CoutSub). I then assigned the Cout into CoutSub, because it is the Carry bit after adding -B to A. For the Overflow, I made a K-map of A,B,Sum and Overflow for substraction and found the Boolean expression for overflow, which is (!ABY) + (A!B!Y).

- Logical and, or, not, xor, xnor, nor

  I assign the output Y according to its respective logical gates according to the function in the mode description, and every Cout and Overflow is 0.

- Binary to one hot

  I set Y into 1, and logically shift that 1 as much as A, so that it would end at its respective position. As it is a one hot representation, exactly one bit is set. So 0 would be represented by 1 at the rightmost position. And 15 would be represented by 1 at the leftmost position. Every Cout and Overflow is 0.

- Assign output to A and B

  I assigned the ouput of Y into A and B, where the Cout and Overflow is 0.

- Find the first one from left

  I used 16 if statements starting from if A[15] = 1, which means checking whether or not there is 1, from the leftmost position accordingly until the last position, or rightmost position. If A[position] is 1, output Y would become that decimal position, and the Cout and Overflow is 0.

4. *Problem faced & how to deal*

- In mode 4, I keep getting wrong answer at first because I forgot to assign the Cout into 0 instead of the Cout I got when adding A and B with the 16 bit Adder. And I forgot to use bitwise not instead of logical not in the Overflow.
- When making the synthesis, I keep getting errors because of my makefile. My macbook keep turning the sensitive makefile into txt which screws up when I tried to simulate at the terminal. Eventually, I borrowed my friend's windows computer to synthesize my code.

5. *Questions for TA*
    - Why does MAC result in so many synthesis and simulation errors compared to windows?