COMP311-1: Logic Circuit Design

Fall 2019, Prof. Taigon Song

Project 2. Due: Dec. 2, 8:59am [Total: 90 + 10 points]

[Your Student ID]

Your Name

No.	Checksheet item	Done? [Y/N]
1.	RCA adder design (no delay, 10 points)	
2.	CLA adder design (no delay, 20 points)	
3.	CSA adder design 1 (no delay, 15 points)	
4.	CSA adder design 2 (no delay, 5 points)	
5.	Four adder design with delay (10 points)	
6.	Tradeoff between area and speed (Synthesis report, 10 points)	
7.	Delay result/analysis (10 points)	
8.	Any other discussion (10 points)	
9.	Bonus: Submission date (+ ____ points)	

1. Goal:

Design and compare four adders: Ripple-carry adder (RCA), carry lookahead adder (CLA), and two carry select adders (CSA). These adders that you design will be modeled with delay so that you can compare the performance between these different adders.

2. Specifications:

The four adders you design will have the following inputs and outputs

- Two 16-bit inputs: ina, inb
- One 1-bit input: cin
- One 16-bit output: sum (testbench wire will have last 4 digits of your student ID)
- One 1-bit output: cout
- All inputs/outputs are assumed to be unsigned decimal

3. Handicap:

Note that (1) the adders that you design are purely combinational, and (2) any top-module/sub-module may not be using 'always', 'assign' for designing your logic. However, any assign statements to re-direct your signal may be allowed (e.g., assign a1 = c1;).

4. Hints and details of the four adders:

4.1. Ripple-carry adder (RCA): Design a ripple-carry adder that you designed for your proj. 1. You are to start from a 1-bit full-adder, then merge it to 4-bit adder, then 16-bit adder. No 'assign' statements or if-else, case statements allowed. Only primitive gates, and modules that you designed can be used. You must understand that this adder is the "ripple-carry adder" and is the slowest among four.

4.2. Carry lookahead adder (CLA): You are to design one 4-bit CLA, then use this CLA to design your 16-bit CLA. For your 16-bit CLA, simply cascade your 4-bit CLAs. Four your information, this project presents you two references.

- https://www.geeksforgeeks.org/digital-logic-carry-look-ahead-adder/

-
https://blog.naver.com/PostView.nhn?blogId=lobdo777&logNo=220433382460&proxyRef

erer=https%3A%2F%2Fwww.google.com%2F

As long as you find a reference that describes a CLA, you are welcome to design based on that reference of your understanding.

Gain understanding of the functionality based these references and design your CLA as the figure below.

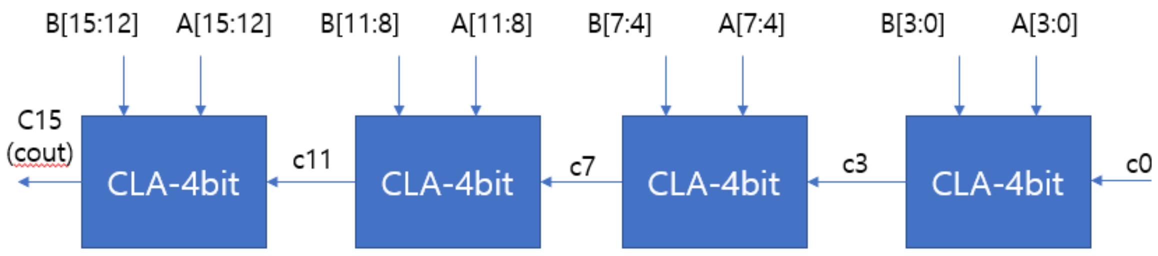

Figure 1 - 16-bit CLA for Project 2

4.3. Carry select adder (CSA-1)

Based on the understanding of following references, you are to design a CSA by recycling your CLA. Please recycle and use two 16-bit CLA to design your CSA. Your MUX may be designed by any format you prefer. Understand why and in what condition CSA provides the best performance in terms of delay. For your reference, below are two references that you could follow:

- https://www.slideshare.net/ABINTHOMAS14/design-and-development-of-carry-select-adder
- https://cms3.koreatech.ac.kr/sites/yjjang/down/dsys10/M05_arith1.pdf

As in 4.2, you are welcome to design your CSA based on a reference of your understanding. (Google will always be your best friend)

Your CSA should look like Figure 2.

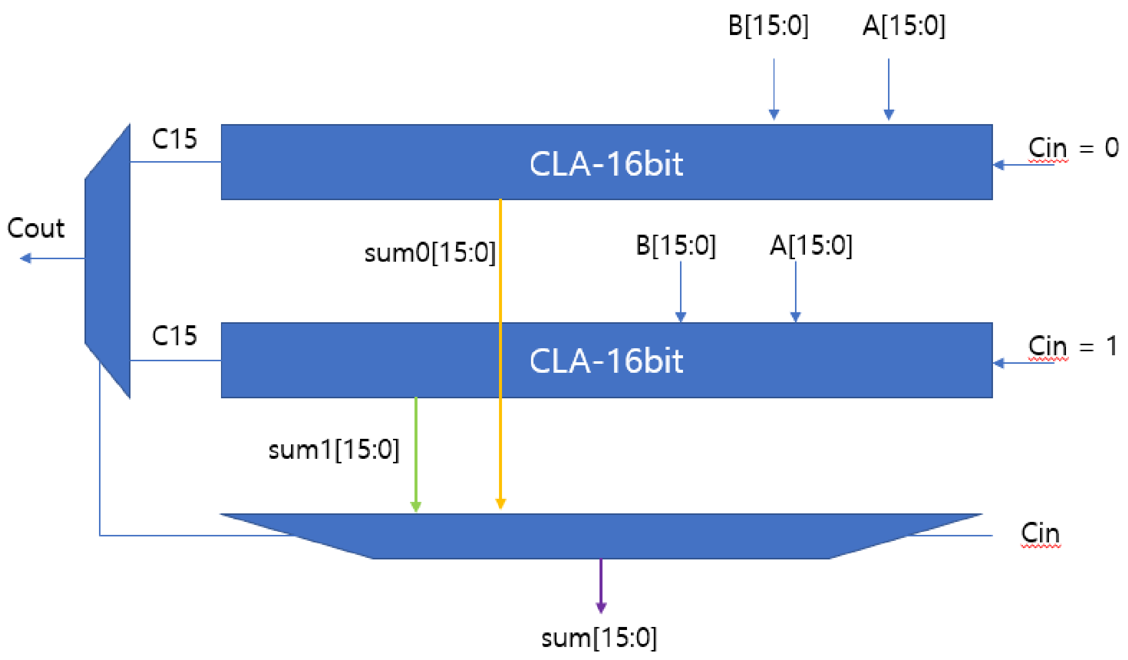

Figure 2– 16-bit CSA for Project 2

4.4. Carry select adder (CSA-2)

Based on the understanding of previous references, you are to design a CSA by recycling your RCA. Please recycle and use two 16-bit RCA to design your CSA (Simply swap two CLAs into RCAs). Think if your CSA-2 will be slower/faster than your CSA-1.

5. Modeling delay

This project studies the performance of these adders by modeling the delay of modules. Below are some info you need to model delay.

- Inverter/Buffer: 3ns
- And/Or: 7ns
- Xor/Xnor: 6ns
- Nand/Nor: 5ns
- MUX: 3ns

6. Testbench design

For your convenience, please use the testbench below.

```
`timescale 1ns/1ns
module adders_tb();
  reg [15:0] in1, in2;
  reg cin1;
  wire [15:0] sum1, sum2, sum3, sum4;
  wire cout1, cout2, cout3, cout4;

  // different adders
  rca16b u1(.a1(in1), .b1(in2), .cin(cin1), .sum(sum1), .cout(cout1));
  cla16b u2(.a1(in1), .b1(in2), .cin(cin1), .sum(sum2), .cout(cout2));
  csa16b1 u3(.a1(in1), .b1(in2), .cin(cin1), .sum(sum3), .cout(cout3));
  csa16b2 u4(.a1(in1), .b1(in2), .cin(cin1), .sum(sum4), .cout(cout4));

  initial begin
        in1 = 0; in2 = 0; cin1 = 0;
    #50 in1[14:0] = $random; in2[14:0] = $random;
    #50 in1[14:0] = $random; in2[14:0] = $random;
    #50 in1[14:0] = $random; in2[14:0] = $random;
    #50 in1 = 16'b1010_1010_1010_1010; in2 = 16'b0101_0101_0101_0101;
       #250 cin1 = 1;
  end

endmodule
```

7. Discussion and Grading

Based on the grading criteria, discuss your results for the below:

- 6. Tradeoff between area and speed (Synthesis report, 10 points)

- 7. Delay result/analysis (10 points)

- 8. Any other discussion (10 points)

Note. Your discussion for 6, 7, and 8 each should be less than 1 page each (font 11).

8. Presentation

At 12/2, Five students will present their report (in 5 minutes) based on a lottery.

(No need for extra preparing)

[Submission]

- Submit your report in ENGLISH.
- Both Online and Offline report.
- Use the first page of Proj 2 for submission. Fill in the boxes based on your completion ratio.
- Submit your report in .pdf and submit your source codes compressed in a .zip file.
- In order to verify if your codes are synthesizable, please attach the synthesis report in your .zip file. (synth_1_synth_synthesis_report_0)
- No need to submit your testbench.
- Early submission: Projects that are submitted before the deadline will get extra credit.
  - ■ 10 points: by 11/22
  - ■ 9 points: by 11/23
  - ■ 8 points: by 11/24
  - ■ …
  - ■ 0 points: by 12/2