ECE 485/585 – Computer Organization and Design
Fall 2022

Project 2 – 32-bit Full Adder Design using VHDL
Report Due: Friday, October 28th, 2022, 11:59 PM

IMPORTANT:	***You must sign and date below acknowledgment statement on the title page of your report.*** **Failing to do so or any violation of this rule will result in an automatic failure of this course.**
Acknowledgment:	I acknowledge all works, including figures, codes, and writings, belong to me and/or persons who are referenced. I understand if any similarities in the code, comments, customized program behavior, report writings, and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action.

I. Introduction

The purpose of this Project 2 is to prepare you for Project 3, in which you will design and implement your custom 32-bit RISC processor, a stripped-down MIPS processor. The goal of this project is to get familiar with VHDL programming, as well as the simulation environment. In this project, you will design and implement a 32-bit adder, one of the basic functionalities of an ALU (Arithmetic Logic Unit).

II. Background

In class, we discussed the 1-bit Full Adder, which consists of 3 inputs (a, b and $Carry_{In}$) and 2 outputs (Sum and $Carry_{Out}$), as shown in Figure 1. The gate implementation of 1-bit Full Adder is shown in Figure 2, and its truth table is shown in Figure 3.

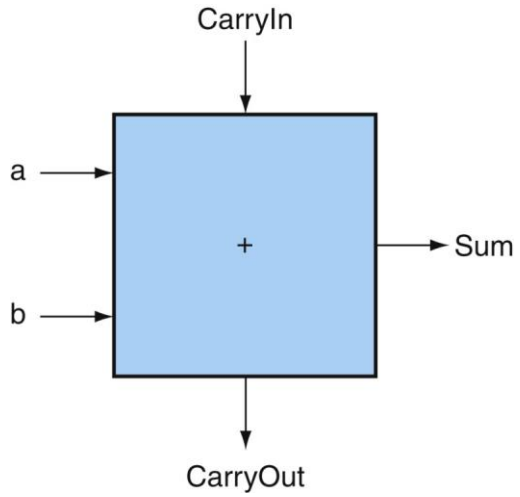

Figure 1. 1-bit Full Adder

ECE 485/585 – Computer Organization and Design
Fall 2022

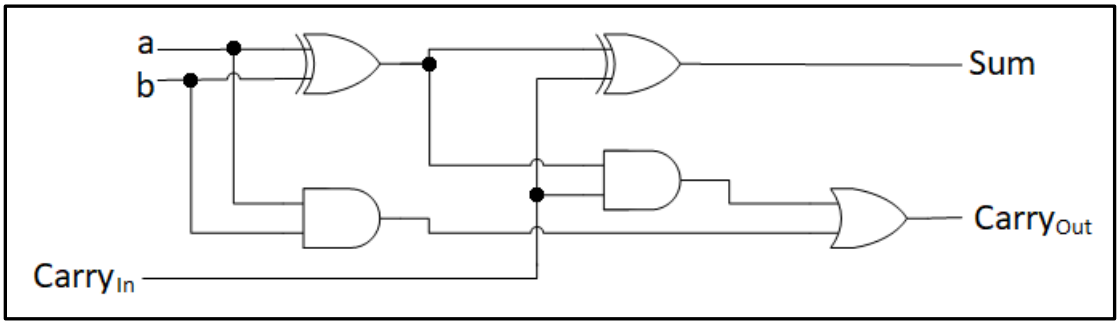

Figure 2. 1-bit Full Adder Gate Implementation

Inputs			Outputs		
a	b	CarryIn	CarryOut	Sum	Comments
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1 + 1 = 11_{two}$

Figure 3. Truth Table of 1-bit Full Adder

A ripple carry adder (RCA) or carry ripple adder (CRA) is a combinational logic circuit that is used to perform the addition of two n-bit binary numbers. RCA requires n-Full Adders in its circuit to add two n-bit binary numbers. For example, if you want to perform the addition of two 4-bit binary numbers, the two outputs from the 1st Full Adder provides the least significant bit (LSB) of the Bit 0 sum (S) and also a carry ($Carry_{Out}$) bit which acts as the carry-in ($Carry_{In}$) bit of the 2nd Full Adder. You can keep adding/cascading more Full Adders to produce any n-bit binary number addition results. Figure 4 shows an example of 4-bit RCA.

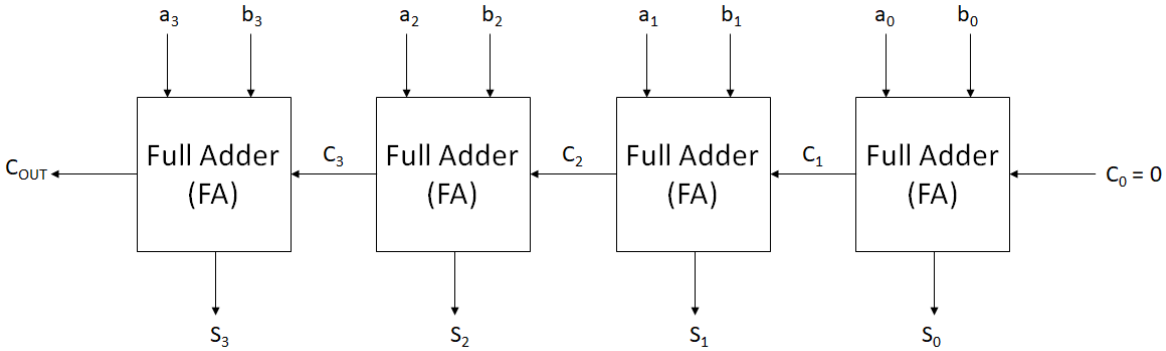

Figure 4. 4-bit Ripple Carry Adder (RCA) using four 1-bit Full Adder

ECE 485/585 – Computer Organization and Design
Fall 2022

As discussed in Lecture Note #6, "ALU Design", a 32-bit ALU can be created by connecting the adjacent 1-bit ALUs in a cascade where the results will ripple from the least significant bit (LSB) *Result0* to the most significant bit (MSB) *Result32*, as shown in Figure 5.

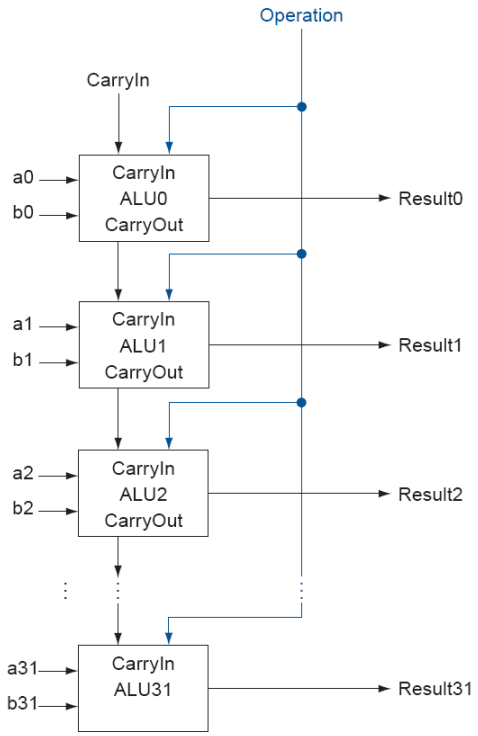

Figure 5. 32-bit ALU

In the RCA, each Full Adder's sum can only be determined after the carry generated by the previous stage Full Adder is produced. Therefore, the complete n-bit RCA result will only be ready once the carry has been rippled through from the LSB Full Adder to the MSB Full Adder. This causes a considerable delay in the n-bit RCA output.

The key to speeding up addition is to determine the carry into the higher order bits sooner so that those carry bits would be ready sooner to produce the sum result faster. For faster addition, Carry Lookahead Adder (CLA) can be used where carries are computed in advance, based on the input values.

The carry logic equation is

$$c_{i+1}=a_ib_i+(a_i+b_i)c_i$$

where a_ib_i is defined as a *generate signal*, and a_i+b_i is defined as a *propagate signal*. Thus, the carry logic equation can be re-written as

$$c_{i+1}=g_i+p_ic_i$$

ECE 485/585 – Computer Organization and Design
Fall 2022

For a 4-bit CLA, carries are calculated as the following:

$$c_1=g_0+p_0c_0$$

$$c_2=g_1+p_1c_1=g_1+p_1(g_0+p_0c_0)$$

$$=g_1+p_1g_0+p_1p_0c_0$$

$$c_3=g_2+p_2c_2=g_2+p_2(g_1+p_1g_0+p_1p_0c_0)$$

$$=g_2+p_2g_1+p_2p_1g_0+p_2p_1p_0c_0$$

$$c_4=g_3+p_3c_3=g_3+p_3(g_2+p_2g_1+p_2p_1g_0+p_2p_1p_0c_0)$$

$$=g_3+p_3g_2+p_3p_2g_1+p_3p_2p_1g_0+p_3p_2p_1p_0c_0$$

By utilizing the above carry logics, a 4-bit CLA can be constructed like in Figure 6.

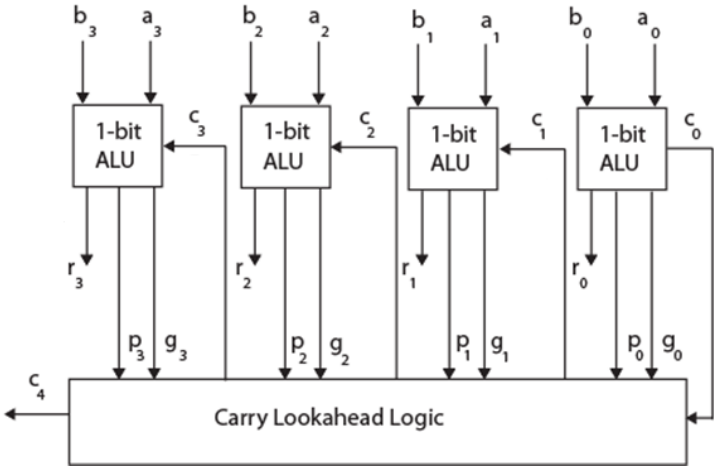

Figure 6. 4-bit CLA

Based on the implementation given in Figure 6, a 16-bit ALU using four 4-bit CLAs can be created cascading 4-bit CLAs where carry ripples between 4-bit blocks as shown in Figure 7.

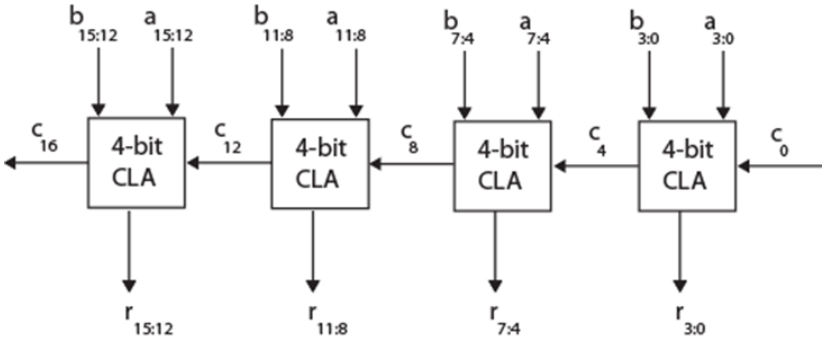

Figure 7. 16-bit CLA

ECE 485/585 – Computer Organization and Design
Fall 2022

III. Design and Implementation

1. Based on the given information, design a 1-bit RCA using the Behavioral Model.
2. Design a 4-bit RCA using Structural Model based on your 1-bit RCA.
3. Design a 32-bit RCA using Structural Model based on your 4-bit RCA.
4. You MUST provide fully commented VHDL Source Codes and your Testbench Codes
5. You MUST provide screenshots of your simulation results for your 1-bit, 4-bit, and 32-bit RCAs using 3 *different two input values of your choice*
   a. *For the 32-bit RCA, you must add the first 4 digits of your A number to the last 4 digits as one of your combinations*
6. Compare the simulation results of your 1-bit, 4-bit, and 32-bit RCA, and discuss them in your report.

IV. Design and Implementation
(For bonus points, ONLY IF completed Part III)

1. Design a 16-bit CLA based on the information given in Figure 6 and 7.
   a. You should start with 4-bit CLA design first
2. Design a 32-bit CLA based on the information given in Figure 6 and 7.
3. You MUST provide fully commented VHDL Source Codes and your Testbench Codes
4. You MUST provide screenshots of your simulation results for your 4-bit, 16-bit and 32-bit CLAs using *3 additional different two input values of your choice*
   a. *For the 32-bit CLA, you must add the first 4 digits of your A number to the last 4 digits as one of your combinations*
5. Compare the simulation results of your 16-bit and 32-bit CLAs, and discuss them in your report.
6. Compare the simulation results of your 32-bit RCA and 32-bit CLA.

V. Project Requirements

1. You are required to design and implement this project **individually or in a group of two**.

If you choose to groupwork, your team member MUST be registered to the same course number!

2. You are required to provide your original VHDL codes and the testbench codes you have used to verify your VHDL codes. Both codes should contain your original comments.
3. You are required to provide simulation results by capturing the screen of the simulation output. Your simulation result must contain the input data used and corresponding

output data. You must clearly state in all captured figures which input data is used and the corresponding output data.

4. I recommend using the same simulator that you have used for Project 1.
5. You must provide results, discussions, screenshots, source codes, testbench codes, and others asked in Section III and Section IV.
6. **Your report should include the following sections:**
    a. Title Page with Acknowledgment and your Signature
    b. Abstract of your report
    c. Introduction *(please remember, introduction and abstract aren't the same)*
    d. Background
        i. Description of your full-adder
        ii. Description of your VHDL simulator and environment
        iii. Anything else that you'd like to address in the background
    e. Simulation Results and discussion
        i. Screenshots of your test cases
        ii. Descriptions of each screenshot
        iii. Discussion of any issues that you faced, any improvements that could be made, etc.
    f. Conclusion
    g. Distribution of work – *if you are submitting this project as a groupwork*
    h. List of references
        i. Write one short paragraph about each of the references and its relevance to completing your project
    i. Appendix
        i. Entire Source Code of your project with comments
        ii. Entire Testbench Codes with comments that used to verify your design
7. The due date is **Friday, October 28th 2022 11:59PM**. *No late submission will be accepted*. You'll need to submit the following package in a single ZIP file to Blackboard.
    a. Your Project Report
    b. Your VHDL Source Codes with your own comments
    c. Your VHDL Testbench Codes with your own comments
8. Refer to the tutorials uploaded on Blackboard if you are unfamiliar with the VHDL environment.