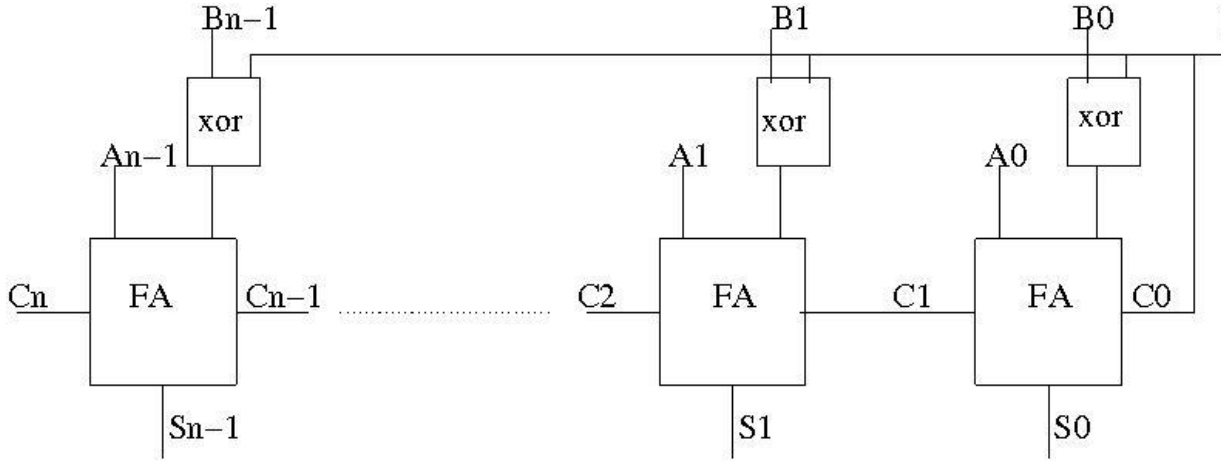
P2

Adders and Subtractors

This lab is going to give you more practice at organising and constructing C++ programs. Its chiefly about constructing functions with clear interfaces and reusing them to build up more complex structures. At the end of this lab you will construct a program that adds and subtracts by modelling a digital circuit, simulating the circuit below with the addition of overflow detection.



Add&subtract with ripple propagation

Schedule

Preparation time	:	3 hours
Lab time	:	3 hours

Items provided

Tools	:	None
Components	:	*None*
Equipment	:	None
Software	:	Eclipse and MinGW

Items to bring

Essentials. A full list is available on the Laboratory website at
https://secure.ecs.soton.ac.uk/notes/ellabs/databook/essentials/

Before you come to the lab, it is essential that you read through this document and complete ***all*** of the preparation work in section 2. If possible, prepare for the lab with your usual lab partner. Only preparation which is recorded in your laboratory logbook will contribute towards your mark for this exercise. There is no objection to several students working together on preparation, as long as all understand the results of that work. Before starting your preparation, read through all sections of these notes so that you are fully aware of what you will have to do in the lab.

> **Academic Integrity** – *If you undertake the preparation jointly with other students, it is important that you acknowledge this fact in your logbook. Similarly, you may want to use sources from the internet or books to help answer some of the questions. Again, record any sources in your logbook.*

Revision History

February 3, 2013	Jeff Reeve (jsr)	First version of this lab created
April 5, 2017	Elena Woo Lai Leng	Supervisor notes
January 31, 2020	Ivan Ling Ting Yang	Updated lab notes

1 Aims, Learning Outcomes and Outline

This laboratory exercise aims to:

- Introduce you to the notion of references
- Enable you to think about how to construct complex code
- Enable you to think how digital circuits might be simulated

Having successfully completed the lab, you will be able to:

- Construct complex code from simple components
- Use the const qualifier
- Use references in functions
- Simulate simple circuits

You are shown below how an adder/subtractor is constructed from basic components (e.g. AND, OR and XOR). The final deliverable of this lab is a program that can simulate the operation of a digital circuit that can add and subtract 8-bit numbers. You will need to think about how you interface you program to the outside world with sensible input and output formats.

---

2 Preparation

Read through the course handbook statement on safety and safe working practices, and your copy of the standard operating procedure. Make sure that you understand how to work safely. Read through this document so you are aware of what you will be expected to do in the lab. The additional work for this exercise takes considerable amount of understanding and preparation. If you wish to attempt the additional work in this lab, make sure you have done enough preparations and reading before the lab session.

Adders

Study the circuits below to appreciate how an adder is constructed and draw figures showing how you will construct the final program. In the diagram below the Sum is the exclusive or of the inputs (~A means not A).

- Write in your logbook codes to implement simple logic gates (AND, OR, XOR) as functions in C++.
- Describe, using suitable flowcharts or codes, how you might create a half-adder, using the functions you have written earlier.
- Subsequently, draw a flow chart showing how you would implement a full-adder and an 8-bit adder with ripple propagation using full adders.

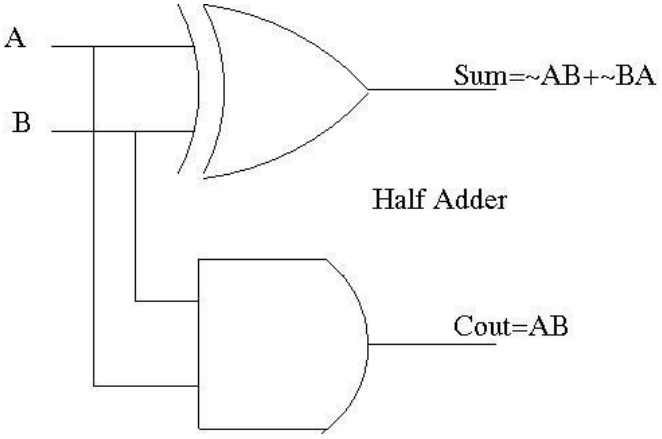


Figure 1: Half adder circuit

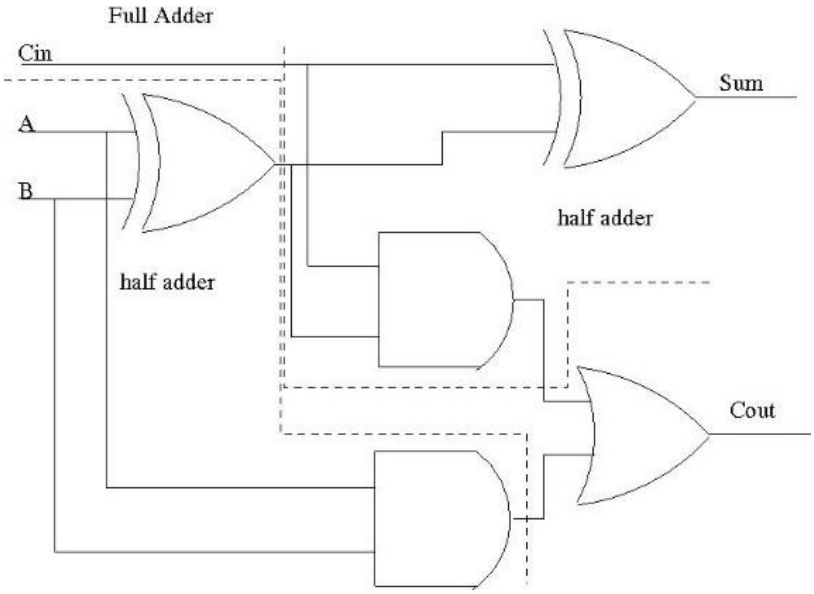


Figure 2: Full adder circuit

---

3 Laboratory Work

The final aim of this lab is to create a main program that integrates the codes written during your lab preparation work into a multi-bit adder with ripple propagation.

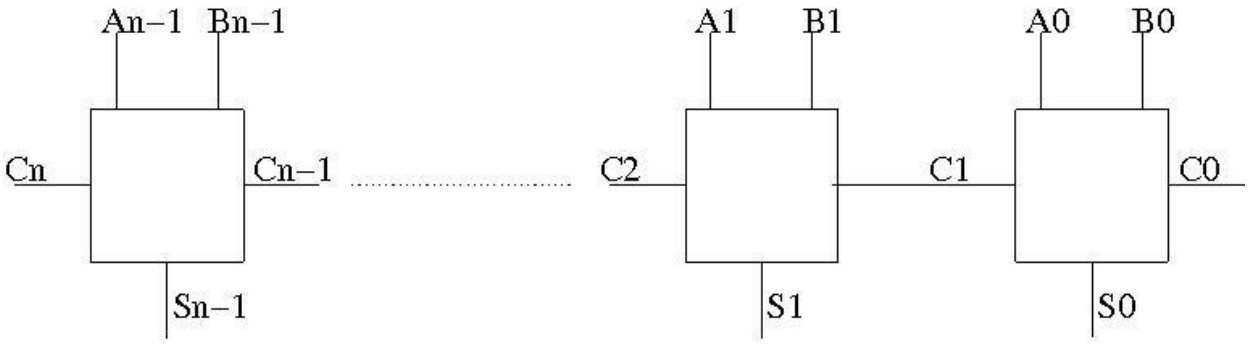


Figure 3: Multibit Adder with Ripple Propagation

3.1 Implement single bit adders

Your program should be able to take a single bit input for A, B and Cin, and give single bit output for Sum and Carry. The function should be implemented using the AND, OR, and XOR functions written in your lab prep.

```
Input your first binary number: 1
Input your second binary number: 0
C_In Value is: 1
Sum is: 0 and Carry_Out is: 1
```

3.2 Implement 8-bit adder

Your program should take in two numbers as inputs and convert it into the appropriate 8-bit inputs required for your multi-bit adder to work. Write your own test program to validate the functionality of your code. The codes should display the following:

```
Input your first number (A): 4
Input your second number (B): 9
Corresponding value A in binary: 00000100
Corresponding value B in binary: 00001001
C_In Value is: 1
00001101
The decimal equivalent is 13
Carry = 0 Overflow = 0
```

3.3 Subtraction

Modify your program such that the user can select between addition and subtraction. Your program should also display the 2's complement result if the user entered a negative number. A sample of the output is as shown:

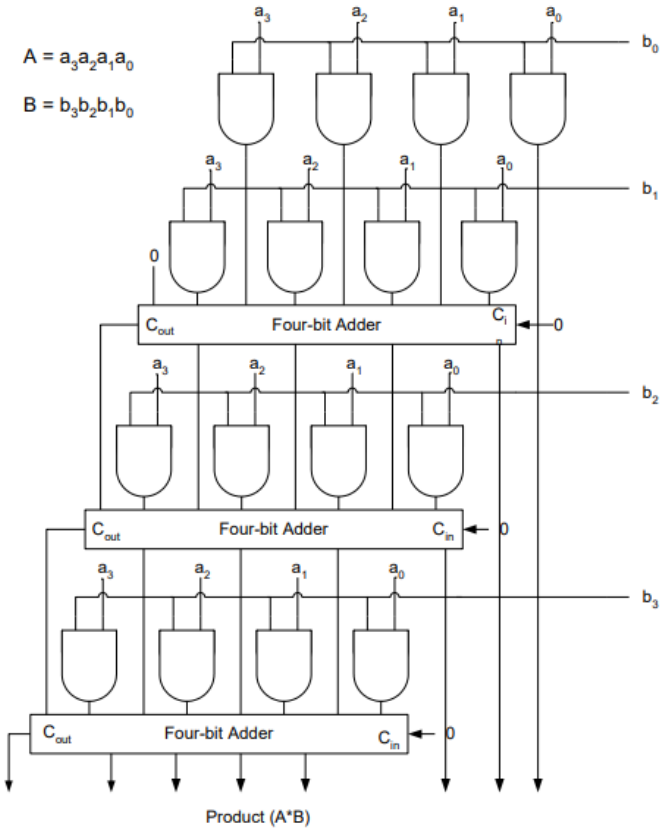
```
Input your first number (A): -6
Input your second number (B): -3
Select operation (+ or -): -
-6--3
Corresponding value A in binary: 11111010
Corresponding value B in binary: 11111101
C_In Value is: 1
Its a negative number, need to convert from 2's complement to decimal.
00000011
The decimal equivalent is -3
Carry = 0 Overflow = 0
```

4 Optional Additional Work

Marks will only be awarded for this section if you have already completed all of Section 3 to an excellent standard and with excellent understanding.

Design and write a program that simulates a multiplier as a cascade of adders.

Below, you will find a diagram showing how a 4-bit adder can be used to implement a 4-bit multiplier.



Appendices

References

HTTLAP

C++ web reference