ENGG2410: Digital Design
Lab 5: Arithmetic Circuits "Adder/Subtractor"
via *VHDL*

Shawki Areibi
School of Engineering, University of Guelph
Fall 2023

Start Date: Week #8 2023
Duration: 1 Week
Report Due Date: Week #9, in the Drop Box

1 Objectives:

- Understand the advantages of using **Hardware Descriptive Languages** for design entry.
- Build several more complex logic circuits and gain increased familiarity with the Xilinx ISE Foundation Tools and VHDL language. Circuits include an adder/subtractor circuit along with components designed in the previous lab.
- Interfacing a 7-segment decoder with the adder/subtractor circuit.
- Show how to describe logic modules and interconnect them in a hierarchical fashion using VHDL.

2 Introduction

By now, you have built combinatorial logic circuits and tested them by using the simulator and by downloading them into the NEXYS 3 Board. It has all been so simple!. So simple, in fact, that you might be asking the question, "Since the computer can figure out the logic design from just the truth table, what do you need me for?" Well, maybe we don't!. On the other hand, we have calculators but we still have accountants. The important fact is that the tedious calculations have been automated, leaving us with the more challenging tasks to perform that computers do not yet do so well. For example, accountants now spend less time adding numbers and more time figuring out neat ways to reduce tax obligations. Likewise, you can spend more time thinking up neat circuits and less time mechanically grinding out the details.

Hierarchical and Modular Design

There is another case where you have to be more involved in the design of the gate-level logic. This occurs when a truth table representation for a design is theoretically possible but impractical. For example, an adder that adds two 8-bit numbers could be specified by a truth table having $2^{16} = 65,538$ entries,

Design Entry Using VHDL

In the 1980's rapid advances in integrated circuit technology lead to efforts to develop standard design practices for digital circuits. VHDL was developed as a part of that effort. VHDL was originally intended to serve two main purposes. First, it was used as documentation language for describing the structure of complex digital circuits. As an official IEEE standard, VHDL provided a common way of documenting circuits designed by numerous designers. Second, VHDL provided features for modeling the behavior of a digital circuit, which allowed it use as input to software programs that were then used to simulate the circuit's operation. In recent years, in addition to its use for documentation and simulation, VHDL has also become popular for use in **design entry** in CAD systems. The CAD tools are used to synthesize the VHDL code into a hardware implementation of the described circuit. In this lab our main use of VHDL will be for synthesis. The tutorials and technical report provided to you with this lab will help you synthesize the required design.

3 Preparation

In the section of lab #5 on the web page you will find the following links:

1. VHDL Tutorial Tech Report "VHDL For Digital Design".
2. Design Entry Using VHDL (SOE).
3. ISE 13.3 VHDL Tutorial (By Digilent).
4. ISE 13.3 In Depth Tutorial (By Xilinx).
5. Controlling NEXYS 3 Display (VHDL CODE).
6. Controlling NEXYS 3 Display (UCF File).
7. VHDL OnLine (Tutorials, Courses, News)

You will use the **VHDL Tutorial "Tech Report"** as a reference to VHDL programming. I assume that you have already read the document **"Design Entry Using VHDL"** in the previous lab.

The Seven Segment Decoder that was built in Lab#4 will be used as a component in this lab. The output of the Adder/Subtractor will be connected to the Decoder to display the result on the 7-Segment display.

3.1 Binary to 7-Segment Decoder

A seven-segment display is often used on computers, watches, VCRs and many electronic devices to display numbers and some characters. It consists of seven independent lights (light emitting diodes (LEDs)) in an "8" configuration as shown below in Figure 1. By turning on different segments, you can display different numbers and some letters.

You should have this component designed and implemented from Lab#4.

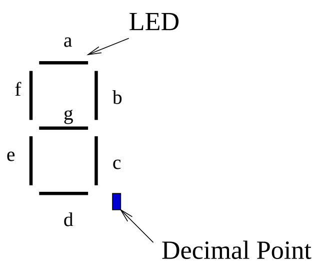

Figure 1: Seven Segment Display.

3.2 4-bit Adder/subtractor

The tutorial of one of the previous labs illustrated a hierarchical design of a full adder. In this lab you will extend the idea of a full adder to design a binary adder-subtractor. Using either the 2's or 1's complement, you can eliminate the subtraction operation and all you will need is the appropriate complementer and an adder (see section 4-4 in your text book). The circuit for subtracting **A - B** consists of a parallel adder with Inverters placed between each **B** terminal and the corresponding full-adder input. The input carry C_0 must be equal to 1. The operation that is performed becomes **A** plus the **1's complement** of **B** plus 1. This is equivalent to **A** plus the **2's complement** of **B**. For unsigned numbers, it gives **A - B** if $A \geq B$ or the 2's complement of **B - A** if $A < B$. You should have a control input to the adder-subtractor to control the operation to be used.

A 4-bit adder-subtractor circuit is shown in Figure 2. Input **S** controls the operation.

1. When **S = 0** the circuit is an adder.
2. When **S = 1** the circuit becomes a subtractor.

Each exclusive-OR gate receives input **S** and one of the inputs of **B**, B_i. When **S = 0**, we have $B_i \oplus 0 = B_i$. If the full adders receive the value of **B**, and the input carry is 0, the circuit performs **A** plus **B**.

When **S = 1**, we have $B_i \oplus 1 = \bar{B}_i$ and $C_0 = 1$. The circuit performs the operation **A** plus the 2's complement of **B**.

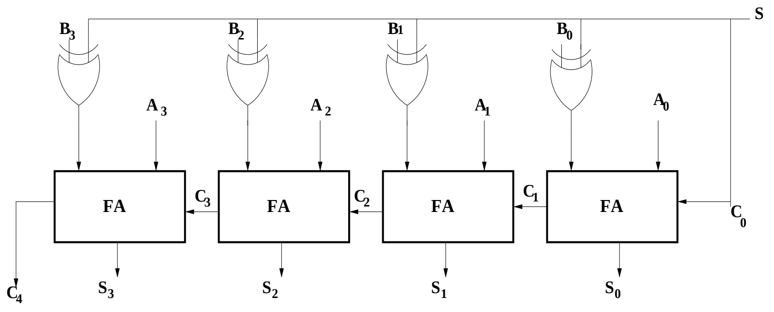

Figure 2: 4-Bit Adder/Subtractor

Figure 3 shows the complete design that integrates the 4-bit adder/subtractor, the seven segment decoder and seven-segment display.

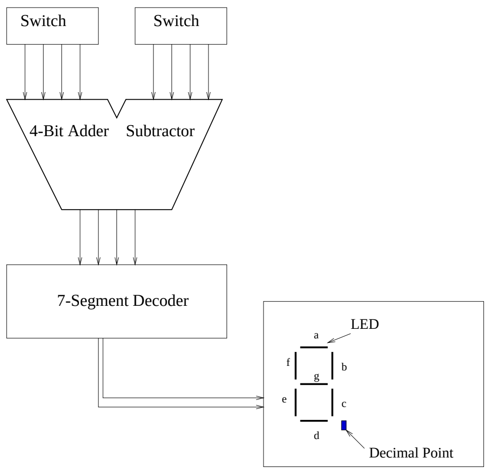

Figure 3: Adder Subtractor Design with Input Switches and 7-Segment Display.

4 Requirements

Before you start the lab you need to be well prepared since this lab is quite lengthy and requires lots of time.

- First, check (make sure) that the 7-segment display on the NEXYS-3 board is working.
- **Reuse** the 7-segment decoder that you built in the previous lab. The VHDL code you prepared in the previous lab will be used and integrated with the VHDL code you prepared for the the adder/subtractor circuit in your final design.
- **Design** the 4-bit adder-subtractor (by simply extending the 1-bit adder that is explained in the lecture notes, tutorial and lab4 tutorial).
- **Connect** the output of the 4-bit adder-subtractor to Light Emitting Diodes (LEDs) on the FPGA board and DEMO the correctness of your design.
- **Interface** the 4-bit adder/subtractor to the 7-segment decoder built earlier and connect the overall design to the 7-segment display as seen in Figure 3 using VHDL.
- Demonstrate your design to the TA.

Important Issues:

- Since you have two operands (4-bits each), you will be using all the available switches on the NEXYS-3 board. Therefore, you can map the S control line to a push button.
- We are assuming that we will be using Binary Numbers in 2's Complement representation.
- The output of the adder/subtractor should be connected to both the LEDS and also the binary-to-7 Segment decoder which will be connected to a single 7-segment display.
- If you are unable to interface your Adder/Subtractor to the 7-Segment Display (for any reason) then you should at least demonstrate the functionality of the adder/subtractor by showing the results on the LEDs. This will enable you to guarantee 80% of the mark for this lab.

5 Deliverables

1. **Lab Preparation File**:
   - Name your file as follows: ENG2410_F23_LAB5_Section(Mon,Tue,Thu,Fri)_Group#_Preparation.pdf
   - The preparation file can be hand written or typed. But has to be neat.
   - The preparation file should consist of the Truth Table, Boolean Equations derived, Circuit Diagram and partial VHDL Code.
   - Only one submission is allowed. Multiple submissions will not be marked.
2. **Lab Final Report File**:
   - Name your file as follows: ENG2410_F23_LAB5_Section(Mon,Tue,Thu,Fri)_Group#_FinalReport.pdf
   - Check the format of the final report below in the next section.
   - Keep the first page only for the title page.
   - Only one submission is allowed. Multiple submissions will not be marked.
3. **Zipped Project File**:
   - Name your file as follows: ENG2410_F23_LAB5_Section(Mon,Tue,Thu,Fri)_Group#_Project.zip
   - The project file should have a README file that explains the switches and LEDs you used in your design.
4. **DEMO**:
   - You will need to demonstrate to the Teaching Assistant that your design fulfills the requirements.
   - The DEMO will take place during the LAB hours and not outside these hours.

6 Report

Below is the general format of the report required:

1. Title Page:
    - Course #, and Date
    - Lab # and name of experiment
    - Your Group #, and Names
2. Start a new page and explain how you implemented your design by providing the following:
    (a) **Problem Statement**,
        i. Briefly describe the problem solved in the lab.
    (b) **Assumptions and Constraints**.
        i. Constraints could be for example using only NAND gates or NOR gates in your design.
    (c) **How you solved the Problem**,
        i. Truth Table.
        ii. Boolean Equations.
        iii. Circuit Diagram.
    (d) **VHDL Code**,
        i. List the VHDL code that you have written.
        ii. Make sure that the code has comments and documented.
    (e) **System Overview & Justification of Design**
        i. Give an overview of the system to be designed.
        ii. Briefly explain how the system works and reasons behind the design.
3. **Circuit Diagram**
    (a) Brief explanation of the hardware.
    (b) The schematic you produced using Xilinx ISE tools.
4. **Error Analysis**
    (a) Printed **Simulation** waveform data, showing that the simulation coincides with the expected truth table of your function.
    (b) Include the VHDL **Test Bench** in an Appendix.
    (c) Describe any problems with the system.
    (d) If no problems in the final system, describe problems/errors encountered during the development and how they were resolved.