Department of Computer Science
Bldg 315
900 Fifth Street
Nanaimo, BC
V9R 5S5
Canada

CSCI 355
Fall Term
Digital Logic and Computer Organization

Lab #5
Combinational Logic with Verilog HDL

Date Due: Saturday, November 2, 11:59 PM **Total Marks: 32**

General Instructions

- **This lab assignment is individual work. You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work. Put your name, student number and instructor's name on the cover page of your report. Failure to follow these conventions will result in a deduction of grades for you. Do not submit folders, zip documents, even if you think it will help.**

- **Assignments must be submitted to VIULearn.**

- **VIULearn will not let you submit work after the assignment deadline. It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.**

- **Partial credit will be given as appropriate, so hand in all attempts.**

1. Objectives

i. Design multibit adder using Verilog HDL
ii. Design multibit comparator using Verilog HDL
iii. Interpret waveforms generated with input/output signals

2. Health and Safety

Any laboratory environment may contain conditions that are potentially hazardous to a person's health if not handled appropriately. Watch for posted information in and around the laboratories, and on the class website.
https://adm.viu.ca/sites/default/files/viu_safety_design_for_facilities_standard_draft.pdf

3. Background

3.1 Multibit-Adder

The full adder adds two input bits (x_i and y_i) as well as a "carry-in" bit (c_i) and generates sum (s_{i+1}) and "carry-out" (c_{i+1}). N-bit full-adders can be used to add two n-bit input numbers $x_{n-1}x_{n-2}\cdots x_0$ and $y_{n-1}y_{n-2}\cdots y_0$ (set $c_0 = 0$). A full adder is shown in Figure 3-1. Multi-bit adder can be constructed by making a chain of full adders as shown in Figure 3-2.

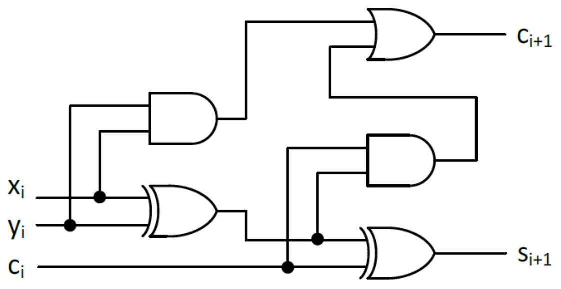

Figure 3-1: Full-Adder

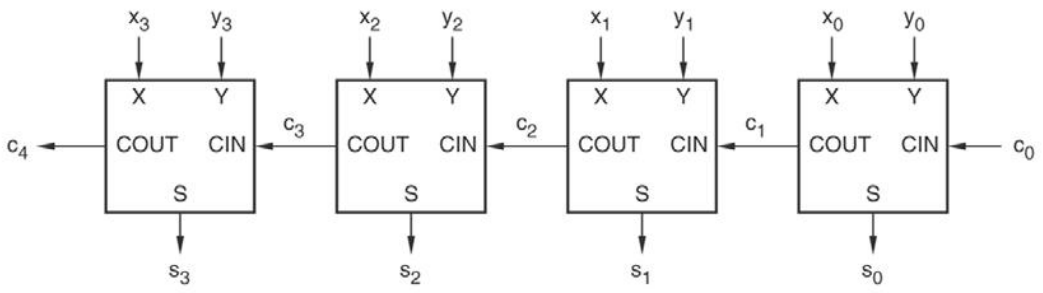

Figure 3-2: 4-Bit Adder

3.2 Multi-Bit Magnitude Comparator using discrete logic

A multi-bit comparator can be constructed using a chain of single-bit comparators (such as constructing a multi-bit adder shown in the previous section). Each comparator must consider both the state from the previous bits as well as its input bits. For example, for the one-bit comparator shown in Figure 3-3:

a) "eq" should be true when "eq_in" is true AND "a = b".

b) "gt" should be true when "gt_in" is true OR "a > b" when "eq_in" is true.

c) "lt" should be true when "lt_in" is true OR "a < b" when "eq_in" is true

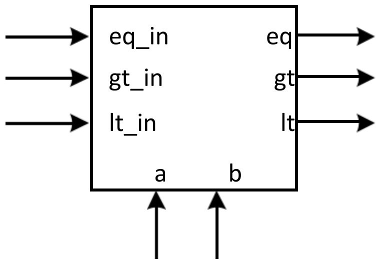

Figure 3-3: One-bit Comparator

The comparators can then be chained together to provide a multi-bit comparator as shown in Figure 3-4.

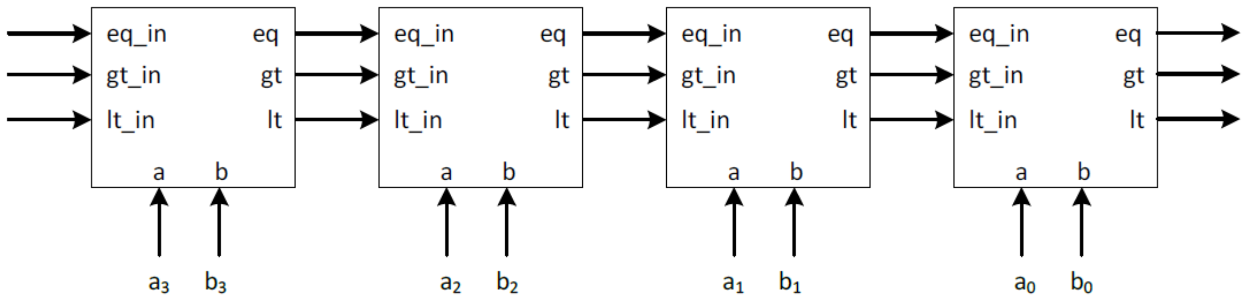

Figure 3-4: 4-Bit Comparator

4. Pre-Lab Tasks

2 **4.1 Why is a multi-bit adder processed from the least-significant bit to the most-significant bit while a multi-bit comparator is processed from the most-significant bit to the least-significant bit?**

2 **4.2 Write the decimal number 7129 in different representations in Verilog HDL:**

Binary = ?
Octal = ?
Hex = ?
Decimal = ?

4.3 Review lecture slides (Week 3) for Verilog HDL.

5. Equipment Required

Cub machines running Linux, Icarus Verilog, GTKWave

6. Lab Procedure

6.1 N-bit adder (Hint: Read Lecture Slides)

5 1 Write the structural (i.e., using gate primitives, see Section 3.1) Verilog HDL code for **a full adder** with the module interface as shown below. Simulate in Icarus Verilog with testbench to verify the operation. You should include the **Verilog main code**, **testbench**, and a picture of the **timing diagram** in your lab report.

```
module fullAdder
(
        input wire ci,
        input wire xi,
        input wire yi,
        output wire so,
        output wire co
);
```

5 2 Write the structural Verilog code for an **eight-bit adder** that uses eight of the full adders from 6.1.1. Simulate in Icarus Verilog with testbench to verify the operation. You should include the Verilog **main code**, **testbench**, and a picture of the **timing diagram** in your lab report.

5

3 Write the behavioral Verilog code for a generic adder (use parameter value 32 for a case of **32-bit adder**) which uses the full adder module from 6.1.1. Simulate in Icarus Verilog with testbench to verify operation. You should include the Verilog **main code**, **testbench**, and a picture of the **timing diagram** in your lab report.

REQUIRED: Demonstrate the waveform to your instructor to prove that your code is working. Make sure that the instructor records that you successfully demonstrated the experiments.

6.2 Four-Bit Comparator

7

1 Write the **continuous behavioural assignment** (i.e., using "assign") Verilog HDL code for the **one-bit comparator** with the module interface as shown below. See Section 3.2 for background specifications about comparators.

Fill out the **truth table for a** one-bit comparator and extract **appropriate expressions**.

Simulate using Icarus Verilog with testbench to verify the operation. You should include the Verilog **main code**, **testbench**, and a picture of the **timing diagram** in your lab report.

```
module comparator
(
input   wire    eq_in,
input   wire    gt_in,
input   wire    lt_in,
input   wire    a,
input   wire    b,
output wire     eq,
output wire     gt,
output wire     lt
);
```

5

2 Write the **structural Verilog HDL code** for a **four-bit** comparator with the help of the one-bit comparator module from section 6.2.1. You may use the module interface as shown below.

Simulate using Icarus Verilog with testbench to verify the operation. You should include the **Verilog main code**, **testbench**, and a picture of the **timing diagram** in your lab report.

```
module fourBitComparator
(
input wire [ 3:0 ] a,
input wire [ 3:0 ] b,
output wire equal,
output wire greaterThan,
output wire lessThan
);
```