

CompEng 2DI4

Digital Logic Design

Laboratory 3:

Programmable Logic

Last Updated: August 11, 2025

McMaster University
Department of Electrical and Computer Engineering
Faculty of Engineering
Hamilton, Ontario, Canada

Copyright © 2023 by Drs. Doyle and Bauman

The manual was prepared to assist in meeting requirements
in the Digital Logic Design undergraduate laboratory.

This Manual is copyright under the Berne Convention. All rights are reserved. Apart from fair dealing for the purpose of private study, research, criticism or review, as permitted under the Copyright Act, 1956, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, electrical, chemical, mechanical, optical, photocopying, recording or otherwise, without the prior permission of the copyright owner.

Trademark Notices:

Microsoft®, Windows 11®, PowerPoint®, Project®, Visio®, Excel® and Word®
are registered trademarks of Microsoft Corporation.

Acrobat® is a registered trademark of Adobe Systems Incorporated.

Avenue® is a registered trademark of Avenue to Learn “Brightspace”.

Trademarks are observed in the text of the report by capitalization of appropriate names.

General Lab Information

Laboratory exercises may offer bonus mark(s) for extra and/or advanced work by the student. The bonus mark(s) will only apply to lab reports that make a clear attempt to meet all non-bonus criteria (including pre-lab). For example, by not attempting to answer a question from the Laboratory Prelab section, the bonus mark(s) will not be considered in evaluation of the exercise. It should also be noted, although the bonuses may make it possible to achieve greater than 100% in the laboratory component of CompEng 2DI4, the maximum grade assigned to the laboratory component will not be greater than 100%.

Labs run every other week of the course. You are required to check Avenue daily to confirm the schedule and receive corrections and/or clarifications to the lab exercises. Pre-labs are due at the beginning of the lab. Laboratory report and code/HDL are due during the student's assigned section. The student is advised that a submission after the laboratory session ends is considered late, no exceptions.

It should be stressed that due to high course enrolment, students shall not be admitted to other laboratory sessions than those assigned. Please enter and exit the lab punctually and pay close attention to the time limits your TAs give you for submitting your lab exercises. Failure to have your lab execution checked because the lab time expired may result in a 0 for the lab – TAs are instructed to not accept/review work that is late. Failure to exit the lab punctually may result in being assigned an automatic zero for the entire lab exercise.

Table of Contents

1.0	Objective and Resources.....	5
2.0	Pre-Laboratory Preparation [20 marks total]	5
3.0	Integrated Circuits	6
4.0	Experiment	6
4.1	The Lamp Controller in HDL	7
4.2	3-to-8 Line Decoder in HDL.....	7
4.3	Binary to Decimal Display Driver in HDL	7
4.4	(Bonus) HDL Functional Simulation of Lamp Controller	7
5.0	Simulating Hardware with Vector Waveform Files.....	8
6.0	Submission Requirements	9

1.0 Objective and Resources

To introduce fundamental concepts of combinational logic and circuit design. To demonstrate the use of a commercial Programmable Logic Device (PLD) design package for schematic and Verilog entry. To introduce the design process for combinational logic in a Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD) device.

This lab introduces hardware description language (HDL) software and design through the integration of programmable logic devices with physical devices, such as switches, buttons, function generators, etc. Looking ahead we will be simulating the physical connections and testing the function of the “programmed” circuit(s) via HDL software. Note section 3.6 has additional information to assist with simulation.

You will need the following resources to complete this lab (**all are FREE**):

1. Software: Quartus Prime Lite Edition 22. Download Quartus Prime Lite Edition 22.1 (<https://www.intel.com/content/www/us/en/software-kit/757262/intel-quartus-prime-lite-edition-design-software-version-22-1-for-windows.html>), install on your personal computer (Windows only).
2. Quartus Prime Introduction Using Verilog Designs. Note that Intel have not updated this manual for Quartus 22.1, but the steps are the same. The document can be found on Avenue with the name `tutorial_quartus18_intro_verilog.pdf` and you can reference chapter 4 onward (note the value for DE10-Lite in Table 1).
3. Reference material about the DE10-Lite available from Terasic ([DE10-Lite v.2.1.0 SystemCD.zip](#))
Also available on Avenue.

2.0 Pre-Laboratory Preparation [20 marks total]

1. Similar to “HelloWorld” when learning a new programming language, like C or Python, you will need to execute a similar simple test of functionality. For our test we will implement a two-way light controller (credit: Altera document “Quartus II Introduction Using Verilog Design”) from Figure 1.

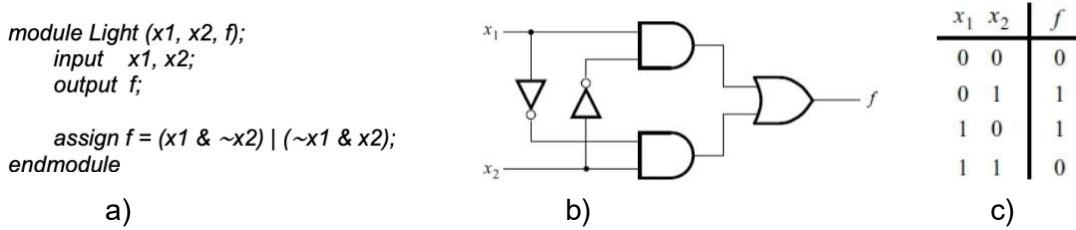


Figure 1: Two-way lamp controller (credit: Altera): a) Verilog code, b) schematic, c) truth table

2. As a new project, implement the two-way lamp controller using Verilog design entry Primitive Gate Modelling as shown in Fig1-a). You will be asked to test this in lab by writing to the FPGA and making necessary connections. [5 marks]

3. As a new project, implement the two-way lamp controller using Verilog design entry Schematic Capture Modelling as shown in Fig1-b). You will be asked to test this in lab by writing to FPGA and making necessary connections. [5 marks]
4. Study the datasheet for the seven-segment-display that will be provided in lab. Design your own custom and minimized combinational logic circuit that takes a 2-bit binary code (in_1, in_0) and drives the display to show the decimal equivalent ($d_6d_5d_4d_3d_2d_1d_0$). For example, an input of 00 (in_1in_0) would display a “0” on the seven segment display (e.g., 00 = 0, 01 = 1, 10 = 2, 11 = 3). Show your design steps. Describe the resultant design using a Verilog modelling method of your choice. [10 marks]

3.0 Integrated Circuits

The following integrated circuits are to be used in this laboratory:

Table 3.1: Integrated Circuits for Lab 3

Identification	Description
SNx4HC	3-to-8 line decoder
MAX10 FPGA	10M50DAF484C7G
Seven segment display	157142V12703
Seven segment display driver	CDx4HC4511, CD74HCT4511

Obtain the data sheets for each of the above devices. Familiarize yourself with their logical and electrical characteristics and bring a copy to your lab session.

4.0 Experiment

Read the following experiment and study the circuits as shown.

REQUIRED: Pre-filling your report with the necessary truth tables, tables, circuit diagrams, etc. and structuring your report such that you only need to record experimental observations will allow you to focus on the experiment(s). Failing to do this will result in an incomplete lab.

NOTE: when including screen captures of design entry, your team member names and student numbers *must* appear in the design entry either as embedded text, not added post image capture. For example, in design entry code this would be in the form of comments, or in schematic this would be using Text Tool.

REQUIRED: ENSURE YOUR MILESTONES ARE VISUALLY CHECKED AND RECORDED BY A TA.

4.1 The Lamp Controller in HDL

Milestone 1: Lamp Controller (onboard LED and switches)

Your TA will ask you to compile in Quartus II and load the CPLD with the lamp controller from either the Primitive Gates Model (AND, OR, NOT, etc.) or the Schematic Capture Model that you completed in the pre- lab. Wire to test.

4.2 3-to-8 Line Decoder in HDL

Milestone 2: HDL Model 3-to-8 Line Decoder (onboard LEDs and switches)

Refer to your textbook for the design of a 3-to-8 Line Decoder (input: 3-bits binary code. output: 1 of 8 bits). For reference, a 3-to-8 line decoder schematic can be found in Mano Figure 4.18 (Figure 4.19 is a 2-to-4 line decoder with enable). HDL Example 4.1 is also a good starting point. Using Quartus II define a 3-to-8 Line Decoder, compile your model and load it on to the CPLD. For inputs connect wires from toggle switches to CPLD inputs. For outputs connect CPLD outputs to LEDs. For your report, include a screen capture of your modelling – ensure all team member names and student numbers appear in the comments of the Verilog model text.

4.3 Binary to Decimal Display Driver in HDL

Milestone 3: Binary to Decimal Display (onboard seven segment display and switches)

Design a minimized combinational logic circuit that takes a 2-bit binary code and drives the display to show the decimal equivalent (e.g., 00 = 0, 01 = 1, 10 = 2, 11 = 3). Carefully note the required input to the seven- segment- display to light an individual segment. Describe the resultant design using a Verilog modelling method of your choice. Compile in Quartus II, load on to the CPLD, and wire to test. For your report include a screen capture of your modelling – ensure all team member names and student numbers appear in the Verilog model screen capture.

4.4 (Bonus) HDL Functional Simulation of Lamp Controller

Bonus Milestone: HDL Simulation of Lamp Controller

Return to your Lamp Controller, but instead of external switch connections you will run a basic simulation by manually creating a functional simulation with Vector Waveform Files or a testbench module to test all input combinations, and perform a functional simulation. Use Primitive Gates Model (AND, OR, NOT, etc.) for your design entry of the lamp controller. Note section 5.0 has additional information to assist with simulation.

For this bonus include in the report:

1. Copy of Verilog HDL code entry define the Lamp Controller (place in appendices) [4 marks],
2. An annotated screenshot of functional simulation results with all combinations of bit inputs. The annotation should illustrate your understanding of the connection between the inputs and outputs. [2 marks],
3. A short description of how you tested and verified the circuit [4 marks].

5.0 Simulating Hardware with Vector Waveform Files

NOTE: This section is informational and can be used to help you with the bonus of Lab 3 and future labs.

HDL software offers the designer the ability to simulate their programmed circuits. With Quartus we have the option of creating a waveform to feed into the input of our new circuits (this is just like manually flipping switches on and off, but instead with software). To do this we can use the waveform generator and manually create waveforms or use the more advanced ModelSim testbench. If you want to pre-test your designs or attempt the bonus milestone then follow the following instructions for waveform simulation.

First you will need to tell the software which device you want to program. If the exact device is not available (MAX10, 10M50DAF484C7G) then assign the device as a generic option (this will result in different properties and pins for the device, but should be functionally the same).

To assign the device, go to “Assignments -> Device”.

Then compile the circuit by going to “Processing -> Start Compilation”.

In order to see how your circuit works, you must perform a simulation. To do this, you need to create a .vWF file that gives the input waveform to your circuit. Go to “File -> New” and choose “Vector Waveform File” under “Verification/Debugging Files”. Save the new file (all filenames must have zero spaces in them).

First set the end time of the simulation by going to “Edit -> End Time” and set it to 100 ns.

Add the nodes by going to “Edit -> Insert -> Insert Node or Bus”. Hit the “Node Finder” button. Then hit “List” to show all the pins in your circuit. Click the double arrows pointing to the right to bring both x and F1 to be selected nodes. Hit “OK”. Change Radix to “binary” then hit “OK”.

You now must supply an input waveform for any of your input pins. In the case of the lamp controller, there is only one input pin, x. Click on it, then choose the blue symbol with a C for “Count Value”. Start value should be 0. Click on the “Timing” tab. A transition should occur every 20 ns. This means it will alternate between values of 0 and 1 over time, so you do not have just a static fixed input. Save the .vWF file.

Next you must associate the .vWF file with your .BDF file. To do this, go to “Assignments -> Settings -> Simulation Settings”. Then for the “Simulation input” box, search and select your .vWF file. Hit “OK”.

Select the tab for your .BDF file. Go to “Processing -> Start Simulation”. Look at your simulation report to see how the output “F1” changes as the input “x” changes. Notice that when the input changes, the output does not change immediately. This is due to the delay inherent in the hardware gate on the device.

Typically you will want to capture a screenshot of the circuit in the .BDF file and the resulting simulation report.

6.0 Submission Requirements

Please ensure you complete the following items prior leaving the laboratory:

1. At the beginning of each lab you are to have your Pre-Lab submitted no later than 2:45pm.
2. Your lab report has a cover page clearly indicating the lab title, date, each member's name, and student number.
3. Cover page must also contain the following statement: As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.
4. Each experiment milestone has been checked by a TA.
5. One report uploaded per 2-member team. Please ensuring both team members have a copy for future reference.

Component	Weight	Grade
Pre-lab	20	
Milestone 1	20	
Milestone 2	20	
Milestone 3	20	
Bonus Milestone	10	
Experiment Observations and Report	20	
Total	100	
Deductions		
Final Score		