CSC258 - Lab 1

Building Circuits using 7400-Series Chips

Winter 2019

1 Learning Objectives

The purpose of this lab is to illustrate the process of building logic circuits by using chips that contain individual logic gates. Although circuits are no longer built this way in industry, it is useful to show how discrete gates are connected together to form a logic function. In future labs, you will be writing code to describe such circuits, but you should always have in mind that the circuits will behave as you see in this first lab. In this lab, you will also gain familiarity with using the schematic builder in Quartus.

2 Marking Scheme

Each lab is worth 4% of your final grade, but you will be graded out of 8 marks for this lab, as follows.

- Prelab: 2 marks
- Part I (in-lab): 2 mark
- Part II (in-lab): 2 marks
- Part III (in-lab): 2 marks

2.1 The Prelab Exercises

All of your lab exercises involve a set of "pre-lab" exercises that are necessary preparation for the lab itself. Unlike other courses, the lab time is meant for the demonstration of your designs, which means that your designs must be complete before attending the lab.

BEFORE each lab, you must read through the sections below and complete all the prelab preparations which are clearly marked in red. Even though you will be working in pairs in your lab session, these prelab exercises are meant to be completely individually, so that both partners are capable of performing the lab on their own, if necessary. These completed pre-lab exercises are submitted electronically on Canvas by the Sunday evening before the lab.

During your lab, use your pre-lab designs to help you complete all the required in-lab actions. The more care you put into your pre-lab designs, the faster you will complete your lab.

3 Introduction to the Digital Lab

This section contains a description of the different pieces of equipment you will use in this lab: the protoboard, the digital switch/light board, the logic probe, the wire strippers and the chip puller.

3.1 Protoboard

The protoboard (aka breadboard) is for holding and connecting chips. As illustrated in Figure 1, chips are inserted across the middle *valley* in the protoboard. The set of holes in a vertical line above the valley are connected electrically, as are the vertically aligned holes below the valley. Therefore, each pin of the chip in the board is connected to the holes above (or below) the pin as directed by the location of the pin with respect to the valley. To make a connection to a specific pin, you need only make connections between the holes, by plugging the bare end of a wire into any of the holes above (or below) that pin.

In the figure, the horizontal lines at the top and bottom of the board delineate holes that are connected horizontally; note that the space in the middle indicates a disconnection. The horizontally-connected holes at the top and the vertically connected holes at the side are usually connected to the power and ground provided by the external connector. The power and ground of the chips are then connected to these strips of holes. The first thing you should do in the lab is connect power and ground to these horizontal and vertical strips. However, do not turn the power supply on until you have verified that all the connections in your circuit are as intended.

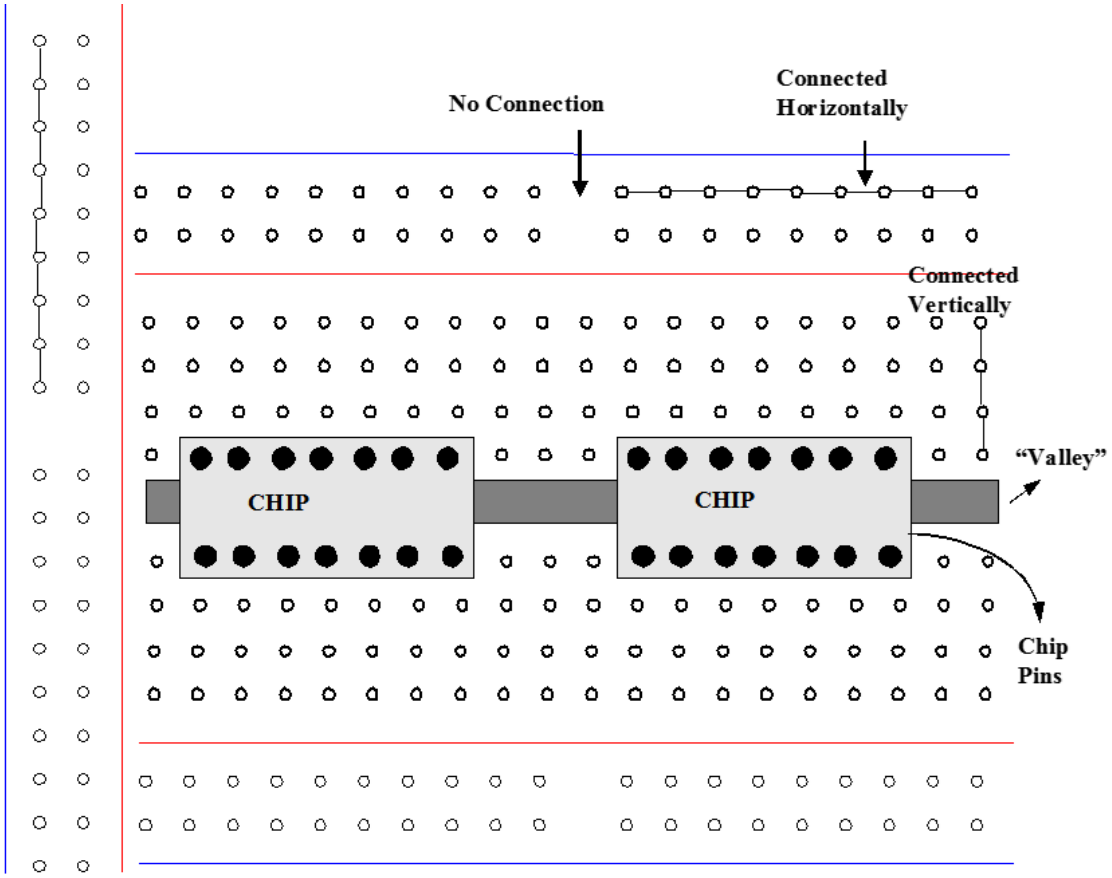


Figure 1: Protoboard

3.2 Digital Switch/Light Board

The digital switch/light board provides switches that have digital output **(5V = logic 1, 0V = logic 0)** and lights that can be driven by logic signals (logic 1 turns a light on, logic 0 turns it off). Test the board by connecting the switches to the lights. The board also provides a clock, which can have its frequency varied by inserting different capacitors into the holes next to it, and a seven-segment display.

3.3 Logic Probe

The logic probe is used for measuring the logic values of signals on the board. Ensure that it has power attached to the correct terminals. To test the probe, touch it to the +5V and ground on the protoboard, to verify that it correctly indicates the values high (1) and low (0) respectively.

3.4 Wire Strippers and Chip Puller

The wire strippers are attached to each workstation to make sure they don't get lost. If you haven't ever stripped a wire, try it!

The chip puller should always be used to remove chips from the protoboard. Doing it with your fingers will bend the pins and ultimately break them, so only remove chips with chip puller!

3.5 7400-series Chip Packages

The chips that you will use in this lab are Small Scale Integration (SSI - meaning there's not much logic on a single chip) 7400 series. Depending on exactly which chip you end up using in the lab you may have to set the logic probe to one of two settings: TTL or CMOS. This setting depends on the type of technology used for the transistors in the chips.

All of the chips you will use are *Dual In-line Packages* or DIPs. Most of the packages have 14 pins, and the pins are numbered from looking at the chip from the top (Figure 2): below the *notch* is pin 1 to pin 7, and above the notch is pin 14 down to 8.

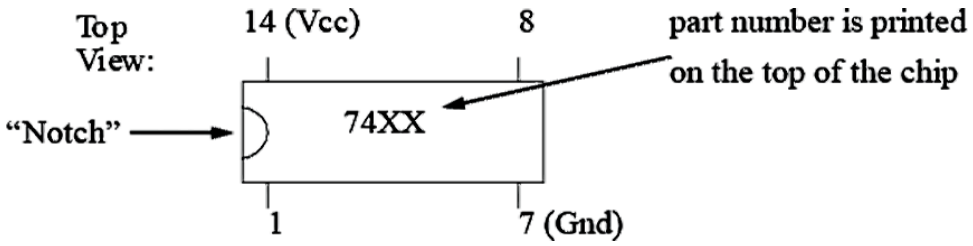


Figure 2: Pin Numbering of TTL Chips

CAUTION: You should always connect the power (VCC, pin 14 in 14 pin chips) and ground (pin 7 in 14 pin chips) of chips to VCC and ground. Leaving the ground pin (pin 7) unconnected is **NOT** the same as connecting it to ground, and your chip will not work correctly.

3.6 Bringing It All Together

The components described above are used to implement the physical circuit that you will be designing below. In the first 15 minutes of the first lab session, the TAs will give a short demo of how to use the protoboard and chips and wires and tools. Before you start working on the lab exercises below though, you should view the module on Quercus called "Lab Breadboard Demo".

This is part of the test of Lab 1. The rest of the lab handout will make more sense to anybody who views this video. However, some people have difficulty reading to this point in the lab handout, and will miss this tip. Some will read this section and decide that they'd rather skip the video and just get to the part that tells them what they need to do. Both types of people will struggle more with the design exercises as a result. The lab handouts contain all the information that you need to complete the labs, and benefit those who have the patience to read and understand everything first before blindly starting to code a solution. The biggest criticism of the course is that the labs are

too hard. They don't have to be, if you take the time to understand them first. And if the solution doesn't jump out at you after reading the handout, come to office hours until it's clear.

4 Preparation Before the Lab

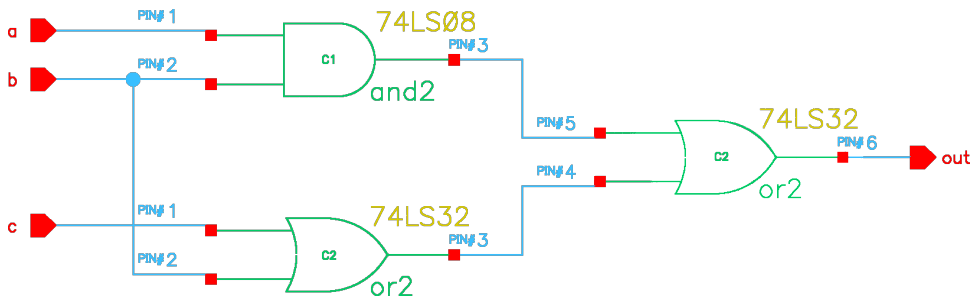
The pre-lab exercises for this lab involve the design of digital circuits using AND, OR and NOT gates in order to create a specified behaviour. However, to make this work in the real world, these gates are built into the 7400-series chips described above, so your circuit diagram also needs to specify how many of these chips you'll need, and how the pins on the chips you use will correspond to the pins on your circuit diagram.

Your task is to design all the circuits in both Parts I and II using **only 74LS04 (NOT), 74LS08 (AND) and 74LS32 (OR) chips**, as given on the attached sheets. Choose the actual pin numbers of the chips that you will use when you build your circuit and show them on your circuit diagram; this will make the construction of your circuit easier.

For example, to implement the following function:

$$f = ab + (c + b)$$

The schematic will look like this:



CHIPS USED:

C1 – 74LS08

C2 – 74LS32

CONNECTED TO ALL CHIPS:

PIN# 7 – Gnd

PIN# 14 – Vcc

Note that you do not need to draw the entire chip; you only need to label which chip you used and which pin number(s) of that chip. Each chip has a unique label, C1 and C2 in this case, and there is a legend to specify the chip type. This will be handy when you have larger circuits where you will have several chips of the same type. We can see that for the AND gate we're using pins 1 to 3 of C1 (a **74LS08** chip), while for the two OR gates we're using pins 1 to 6 of C2 (a **74LS32** chip). The power and ground connections are shown separately. The number two after each gate name (e.g., and2) is not a unique identifier but rather specifies the number of this gate's inputs.

In each case, show all of the steps required to go from the specification given below to the final circuit, including: assigning names to input and output wires, deriving a truth table, the logic function, and then a schematic picture of the final circuit, with pin numbers and chip types.

Important: You are allowed to use only the following packages (see Figure 3): 74LS04 (NOT gates), 74LS08 (AND gates) and 74LS32 (OR gates).

5 Part I

The multiplexer is a device that selects one of multiple inputs to be output. The following Boolean function is a 2-to-1 multiplexer.

$$f = xs' + ys$$

Note that s' is the notation for "s inverted" or "not s". As we can see, when the select signal s is 0, the signal x is shown at the output. However, when s is a 1, the y signal will appear at the output. This is an extremely useful circuit with multiple applications such as in a datapath of a CPU which you will be implementing in one of the future labs.

Perform the following steps:

1. Draw a 2-to-1 multiplexer design using the gates specified above. Indicate pin numbers on the chips. You do not need to draw the entire chip, but you must specify which chip was used for which gate and which pins the inputs and outputs are connected to. Show this design to your TA as part of your prelab to verify that the design is correct before you start to implement it. **(PRELAB)**
2. Write out the truth table for the design and show that to the TA as well as part of the prelab. **(PRELAB)**
3. Wire your design on the protoboard and demonstrate the functionality to the TA. Your results should match your truth table from the prelab.

6 Part II

Build the gate-level implementation for the following Boolean expression:

$$f = (a + b)' + cb'$$

Perform the following steps:

1. Draw the function shown above using the gates specified in the lab preparation. Indicate pin numbers on the chips. You do not need to draw the entire chip, but you must specify which chip was used for which gate and which pins the inputs and outputs are connected to. Show this design to your TA as part of your prelab to verify that the design is correct before you start to implement it. **(PRELAB)**
2. Write out the truth table for your design and show it to the TA as another part of the prelab. **(PRELAB)**
3. Wire your design on the protoboard and demonstrate the functionality to the TA. Your results should match your truth table from the prelab.
4. Is there a cheaper implementation for your design, assuming you are still limited to using the same three chip types? If yes, explain by rewriting the boolean function. For this question we consider a given implementation cheaper if it uses fewer gates or if it uses the same number of gates but fewer chips. **(PRELAB)**
5. Once the final circuit is working and you have shown it to your TA, the TA will intentionally introduce a bug into the circuit. Your task is to find the bug in the circuit by methodically analyzing the circuit behaviour. Do not try to just guess where the bug is. Instead, use the logic probe to test each part of the circuit. Once you find the bug, show the working circuit to your TA and explain how you found the cause.

7 Part III

In this section, you will start on your first Quartus Prime project using the schematic builder. You will design Part I and verify its functionality with the truth table from your prelab.

Perform the following steps for the logic expression given in Part I and test the circuit on the DE1-SoC board.

1. Open Quartus Prime (version 18) and go to File > New... and select New Quartus Prime Project.
2. Click Next and under **Directory, Name, Top-Level Entity** select your working directory and type the name of your project. The project name can be anything, but you should use meaningful names that make it easy to identify the project in the future. Do not use space characters in the project name. The top-level design will automatically fill out to be the same name as your project.
3. Click Next until you reach **Family & Device Settings** and select the chip 5CSEMA5F31C6 under Available Devices and then click Finish. This device belongs to the Cyclone V FPGA family and is the device present in the DE1-SoC board in your lab stations. Selecting an incorrect device is a common source of errors. If you notice that the device name shown under Hierarchy in the Project Navigator in the left side of your Quartus window is incorrect, right click on the device name and select Device.. to change it.
4. Click File > New... again and select Block Diagram/Schematic File. This should automatically open a schematic view window.
5. To place an object click on Symbol Tool (shown as an AND gate) and expand the library until you get to primitives > logic, where you can find all the gates you might need. The number next to each gate name denotes the number of gate inputs. Place all the gates you need.
6. You will also need to define the inputs and outputs of your circuit. Use the drop down Pin Tool to the right of the Symbol Tool to place three inputs and one output symbol.
7. To connect the symbols together, use the Orthogonal Node Tool (shown as a thin 90 degree corner). Make sure there are no x markings shown in the schematic after you do that. To delete any misplaced wires, right click on them and select Delete.
8. Obtain a copy of the `DE1_SoC.qsf` file available under Labs on Portal and place it in your design directory. This file associates signal names to pins on the chip. If you use these exact signal names for the inputs and outputs in your design, the tool will connect those signals to the appropriate pins. You can examine the file in an editor to see the names and pin numbers. Note that pin numbers will not seem very meaningful to you. These pin numbers are provided by the manufacturer of the board in documentation. The `DE1_SoC.qsf` file was created by consulting this documentation.
9. Click on Assignments > Import Assignments... and import the `DE1_SoC.qsf` file.
10. If you open Assignments > Pin Planner, you can see all the assignments of signal names to pin numbers (e.g., SW[0] to pin number PIN_AB12).
11. Name the inputs of your design with *SW[0]*, *SW[1]*, and *SW[2]* and your output as *LEDR[0]*. You can do this by double-clicking on the input and output symbols. This process will allow you to use 3 switches to provide inputs to your circuit and observe its output on the first red LED.
12. Once you have completed your design, click Processing > Start Compilation.
13. When compilation is done, click Tools > Programmer and a window will appear.
14. Go to Hardware Setup and ensure Currently Selected Hardware is DE1-SoC [USB-x] and close the window.
15. Click Auto Detect and select *5CSEMA5* and click OK.
16. Double click *<none>* for device *5CSEMA5* and load SOF file (usually under folder "output_files") and device will change to *5CSEMA5F31*.
17. Ensure Program/Configure for device "*5CSEMA5F31* is checked and click Start.
18. Verify that your design is correct by matching it to the truth table from your prelab.
19. Demonstrate the circuit to your TA to receive the final mark for the lab.

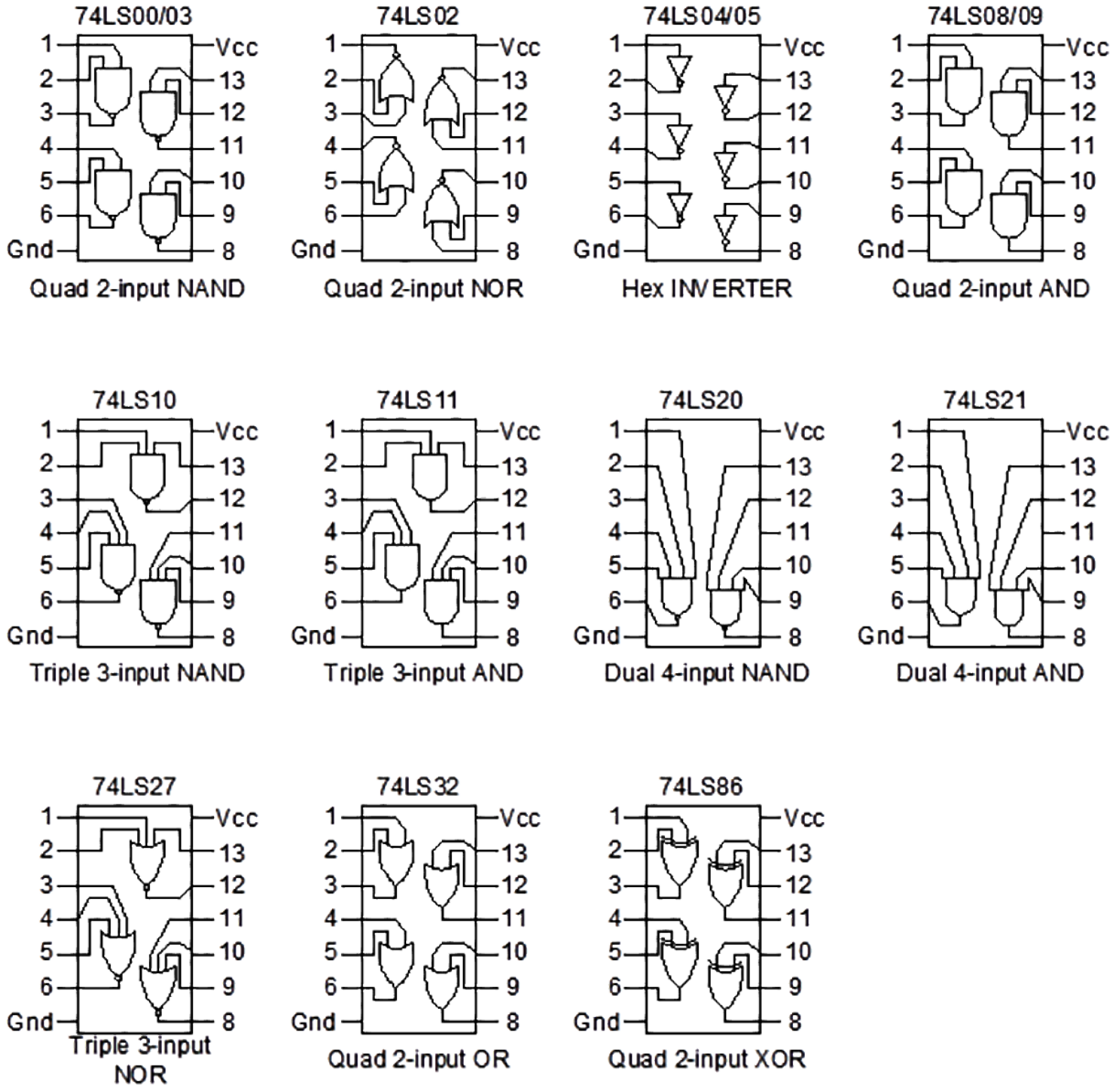


Figure 3: Schematics of 7400-series TTL Chips

Digital Board Header Pin Assignment					
Pin#	**Description**			**Description**	**Pin#**
1	Switch #1	o	o	Switch #2	2
3	Switch #3	o	o	Switch #4	4
5	Switch #5	o	o	Switch #6	6
7	Switch #7	o	o	Switch #8	8
9	Ground	o	o	NC	10
11	Ground	o	o	NC	12
13	Ground	o	o	NC	14
15	Ground	o	o	NC	16
17	LED #1	o	o	LED #2	18
19	LED #3	o	o	LED #4	20
21	LED #5	o	o	LED #6	22
23	LED #7	o	o	LED #8	24
25	Ground	o	o	NC	26
27	Ground	o	o	NC	28
29	Ground	o	o	NC	30
31	Ground	o	o	NC	32
33	Clock	o	o	NC	34
35	NC	o	o	NC	36
37	NC	o	o	Pulse Button	38
39	NC	o	o	NC	40

Figure 4: Digital board header pin assignment

Appendix: Access to Quartus and Modelsim

A Access to Quartus and Modelsim outside of Labs

You will need to use two tools, **Quartus** and **Modelsim**, in this course to complete some of your prelabs. Quartus is the main software tool you will be using to program the FPGA hardware, while Modelsim is the simulator you will use to test the functionality of your hardware design and debug your circuit before programming the board.

Here are different ways that you can have access to these tools:

1. Both Quartus Prime Lite version 18.1 and ModelSim are available in the Computer Science Teaching Laboratories (formerly known as CDF). You can launch them from a terminal by using the `quartus` and `vsim` commands respectively.
2. If you are connecting to the CDF server remotely (via ssh) you need to have X forwarding enabled as both these tools have a graphical interface. You can do this as follows:

   `ssh -Y your_username@teach.cs.utoronto.ca`

   Note that the student usernames on CDF have recently changed to be your UTORids. For more remote access options, please see:

   `http://www.cdf.toronto.edu/using_cdf/remote_access_server.html`
3. You can also (and highly encouraged to) install the free version of Quartus, *Quartus Prime Lite Edition*, on your own laptop/desktop. Before you start, please check the system/memory requirements at: `http://fpgasoftware.intel.com/requirements/18.0/` and read the caveats below.

   Here are the steps to install Quartus:

   (a) You can download **Quartus Prime Lite Edition 18.0** via the following link: `http://fpgasoftware.intel.com/?edition=lite`. This download also includes **Modelsim-Altera**.

   Downloading the combined version of the files is easier, but if you want to go the "Individual Files" route (*e.g.* to save some disk space), you can uncheck all the device families **except** Cyclone V (see Figure 5).

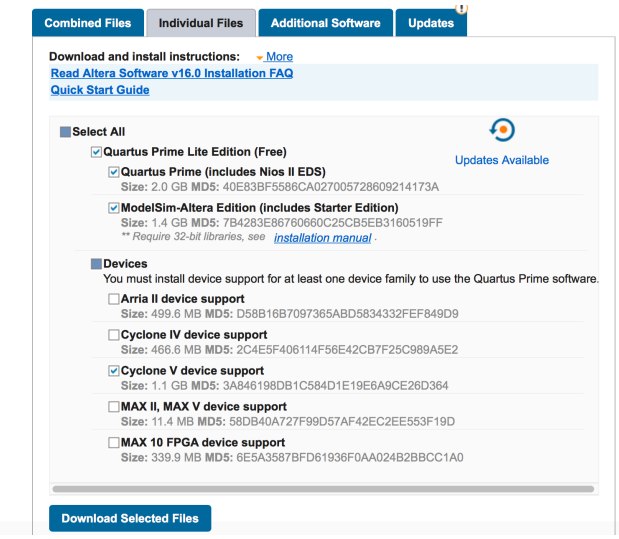


Figure 5: Minimum subset of Quartus that you need to install

A.1 Caveats

Quartus is currently supported on Windows and Linux. If you have a Mac, you would need to install and run Quartus on a virtual machine. Here are some pointers, if you want to do that:

- Before you start, make sure you have enough disk space in your vitual machine to install Quartus. The Quartus installation requires ∼13GB of storage, so you'd probably need your virtual disk to have 30-35GB of storage or so.
- Install VirtualBox from here: `https://www.virtualbox.org`. It is a well-known powerful virtualization software that is completely free.
- If you are considering running Windows on a virtual machine on your Mac, you need a Windows license. You can get one *free* Windows license as a students in Computer Science, Engineering and iSchool, while attending classes. For more information, visit the Microsoft Imagine webstore at `http://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=30b88fca-b08b-e011-969d-0030487d8897&vsro=8`. The "How do I get access?" link explains what you need to do to get an account. I will provide a CD key and an ISO file to install windows.
- If you only need to have Windows to run Quartus, then you can also use Microsoft Edge program virtual machines that you can download for free from here `https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/`. Just note that those VMs expire after 90 days, which is almost enough for this semester.
- Another possible option is to install a Linux distribution (such as Ubuntu `https://www.ubuntu.com/` or Mint `https://www.linuxmint.com/`), which are completely free!
- Now you can install Quartus on your virtual machine.

A.2 Buying DE1 Board

Lastly, if any of you are thinking of buying a DE1-SoC board (which is not necessary or required for this course, and is completely optional), then you could get one from Terasic with a UofT discount. You can find more details here: `http://www-ug.eecg.toronto.edu/desl/handouts/buying_de1.html`. Just be sure to check with them how long it will take for them to send it to you and you receive it.