Laboratory Exercise 1

Building Circuits using 7400-Series Chips

Revision of September 11, 2024

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. This is also a useful skill to have when making projects using boards such as Arduino, which often use a breadboard and discrete components.

In future, you will write System Verilog code to describe circuits, but **you should always have in mind that the circuits will be constructed as you see in this first lab**. This mindset is critical to being successful at writing **working** System Verilog code for hardware.

This lab exercise will have three components:

1. **Pre-lab Online quiz**: There is a short 2-question quiz each student must complete before coming to your lab 1 session. The quiz can be found here.
2. **Pre-lab preparation**: You will design some circuits and draw the schematics prior to coming to lab. You should be prepared to show/explain your schematic to your TA in lab.
3. There will be an in-lab component where you will build the circuits you designed in the first component, but now using the actual physical 7400 chips. **You will have to worry about applying power to the real chips, i.e., make sure each chip has a connection to 5V and GND. If your circuit is not working, that's the first thing to check.**

1 Lab Books

We encourage you to do your preparations in a **lab book**. This is good practice and will help you keep organized. In industry, you are often asked to keep your notes in lab books because this helps to document your work for various kinds of reporting and as evidence when filing patents where it is important to show dates and times of when ideas are conceived. This does not need to be a new book; feel free to use an existing lab book or keep an electronic notebook. What is important is that your notes are clear and legible and you can show and explain your work to a TA if asked.

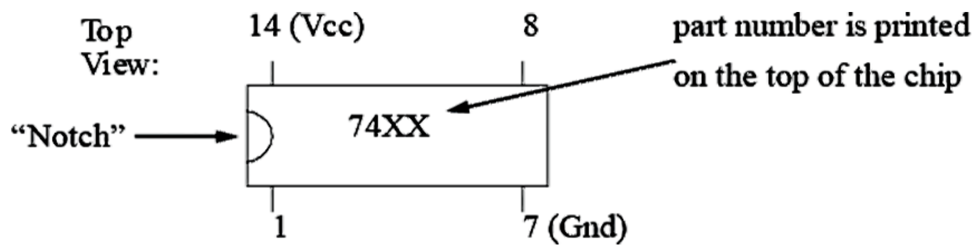


Figure 1: Typical DIP Package

2 The Equipment

This section provides an overview of the hardware that you will use to complete the in-lab portion of this lab assignment. You will build circuits using physical 7400 chips in the lab on a breadboard. We begin with a description of the different pieces of equipment you will use: the 7400-series chip packages, breadboard, logic probe, wire strippers and digital switch/light board.

Note: Please make sure to watch the accompanying video *prior to coming to lab*, which offers additional guidance on using the equipment discussed below. The video can be found **here**.

2.1 7400-series Chip Packages

The chips that you will use in this lab are Small Scale Integration[1] 7400 series devices.

All of the chips you will use are *Dual In-line Packages* or DIPs. Most of the packages are 14 pins, and the pins are numbered by looking at the chip from the top: Holding the chip with the notch on the left end, the pins below the notch are 1 to 7, and the pins above the notch are 14 to 8. NOTE: Pin 14 must always be connected to VCC (+5V) and pin 7 to ground (0V). Figure 1 shows the layout of a typical DIP package.

2.2 Breadboard

The breadboard is for holding and connecting chips. As illustrated in Figure 2, chips are inserted across the middle *valley* in the breadboard. The set of holes in a vertical line above the valley are connected electrically, as are the vertically aligned holes below the valley. So, each pin of the chip in the board is connected to the holes above (or below) the pin. To make a connection to a specific pin, you need only make connections between the holes by

[1]SSI - meaning there are not many transistors in a single chip.

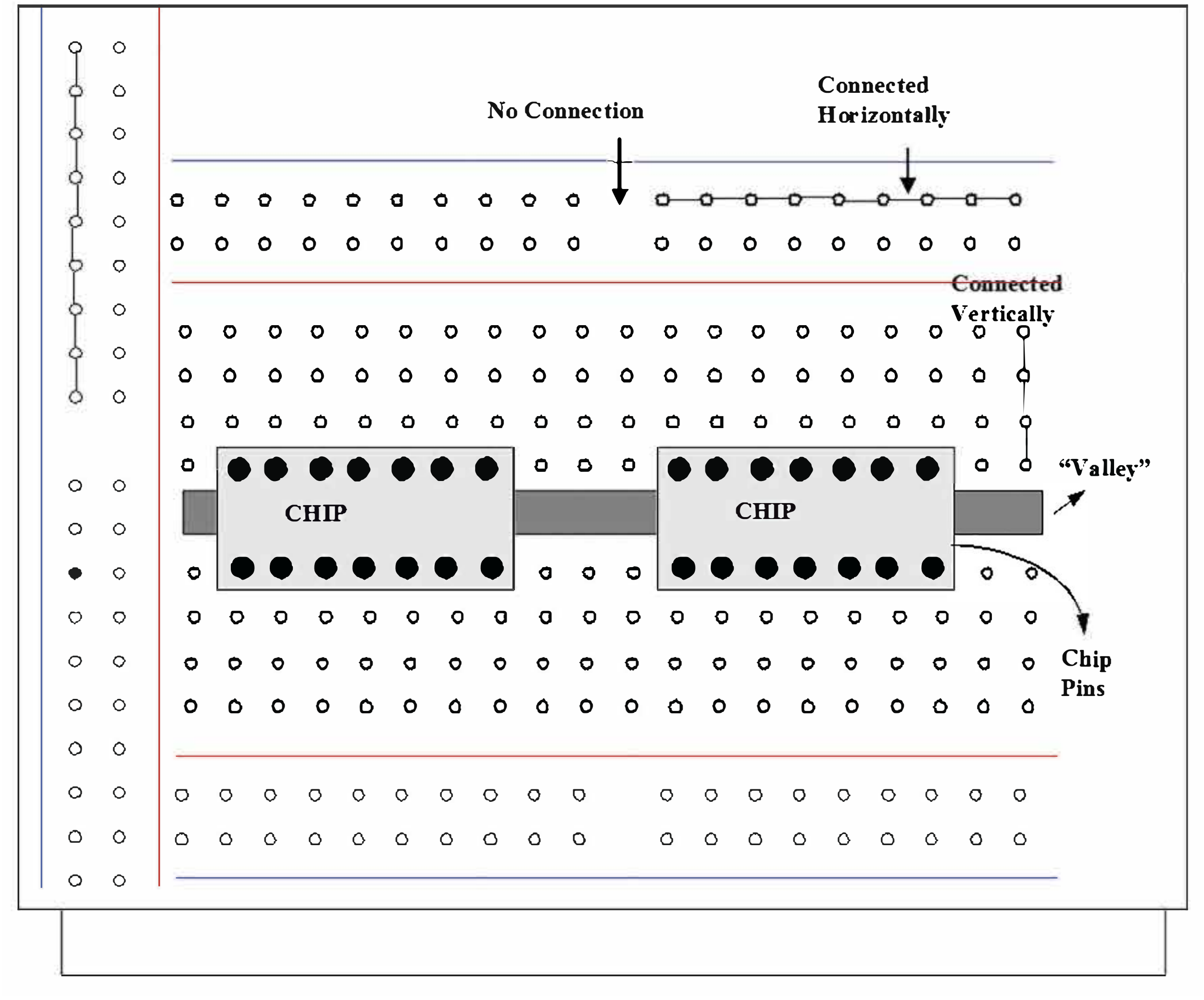


Figure 2: The Breadboard

plugging the bare end of a wire into the holes above or below the pins.

In Figure 2, 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.

2.3 Digital Switch Board

The digital switch 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). The switch board can be connected to the breadboard using the header pins, as explained in the **video** referenced earlier in Section 2. Figure 7 illustrates the board's header pin assignment. Following the video, test the board by connecting switches to the lights via the Header Pin connected to the breadboard.

2.4 Logic Probe

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

Unlike CMOS chips, which use MOSFET transistors, the 7400 series chips use TTL (Transistor-Transistor Logic) with bipolar junction transistors. This distinction is important to us because the logic probes in the lab can switch between TTL and CMOS settings. Make sure to select the TTL setting when probing the 7400 series chips.

2.5 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 breadboard. Doing it with your fingers will bend the pins and ultimately break them, so please don't do this!

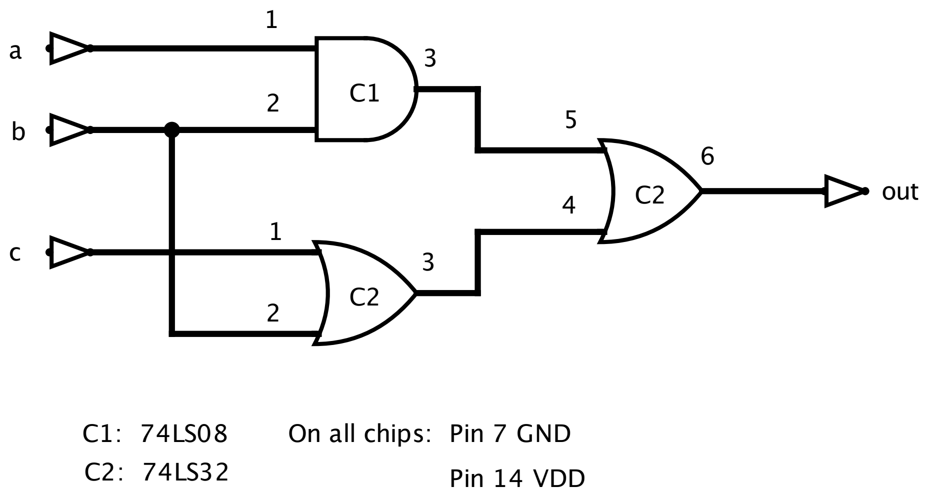


Figure 3: Example Schematic Diagram

3 Preparation

Design all of the circuits in both Parts I and II using **only 7404 (NOT), 7408 (AND)** and **7432 (OR)** series chips, as shown in Figure 6. Choose the actual pin numbers of the chips that you will use when you build your circuit and show them on your circuit diagram, i.e., *schematic diagram*. This will make the construction of your circuit easier.

For example, to implement the following function:

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

the **schematic diagram** is shown in Figure 3. Check the correspondence of the pin numbers on the schematic with the pin-out information sheet in Figure 6.

If you want to see the schematic for the DE1-SoC board we will be using in the lab, you can find it **here**. Every chip and component on the board and all of the connections are shown on the schematic. A board designer will use the schematic to physically lay out the board and draw where the wires should go on the board.

Note that you do not need to draw the entire chip; you only need to label which chip you used and which pin numbers on the chip you used. Each chip has a unique label, C1 and C2 in this case, and there is a legend to indicate the type of chip. This will be handy when you have larger circuits where you will have several chips of the same type. We can see that the AND gate (C1) uses one **7408** chip using pins 1 to 3 and one OR gate (C2) **7432** chip using pins 1 to 6. The power and ground connections are shown separately.

In each part below, show all of the steps required to go from the specification given to the final circuit, including: assigning variable names to inputs and outputs, deriving a truth table, the logic function, and then a schematic diagram of the final circuit, with pin numbers and chip types.

Important: You are allowed to use only the following packages: 7404 (NOT gates), 7408 (AND gates) and 7432 (OR gates). Figure 6 shows the pin-outs for these chips.

Steps 1 and 2 in Part I and Part II below should be completed prior to lab as preparation. Your TA will ask you questions about your preparation as part of your in-lab assessment.

4 Part I

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

$$f = x\overline{s} + ys$$

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 show at the output. This is an extremely useful circuit with multiple applications such as in a datapath of a CPU that you will be implementing a part of in the future labs.

Perform the following steps.

1. Draw the 2 to 1 multiplexer schematic diagram using only the gates specified in Section 3.
2. Write out the truth table for the design.
3. Build your circuit using 7400 series chips and demonstrate your working circuit to your TA. **This step will be done in-lab.**
4. Is there a cheaper implementation for your design, i.e., using fewer chips?

5 Part II

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

$$f = \overline{a}b\overline{c} + \overline{b}(\overline{a}\ \overline{c} + ad) + \overline{a}\overline{b}d$$

Perform the following steps, but **first read through all steps** as you may be able to save some time and effort after reading Step 5 .

1. Draw the schematic diagram for the function shown above using only the gates specified in Section 3.
2. Write out the truth table for the design.
3. Verify that your circuit matches your truth table.
4. Build your circuit using 7400 series chips and demonstrate your working circuit to your TA. **This will be done in lab.**
5. Is there a cheaper implementation for your design, i.e., using fewer chips? If you can find a cheaper implementation, show the Boolean Algebra used to arrive at the simpler equation. You may then carry out Steps 1-3 with the simpler implementation, instead of the original form.

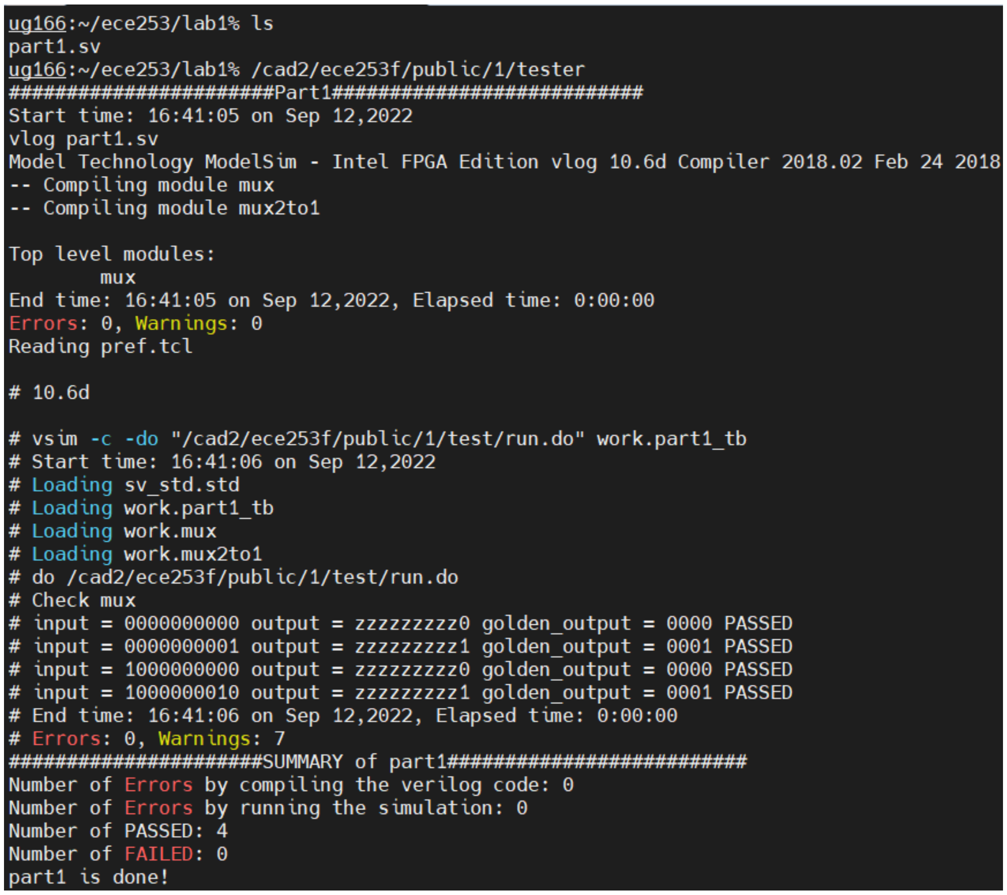
6 Submission

For every lab, this section will describe exactly what you must submit for each part. First, read through the submission instructions PDF provided on Quercus here; you can also refer to the tutorial videos here.

For lab 1, you must just submit a System Verilog file (`mux.sv`) that is already provided for you. This is to teach you how the submission process will work for all the other labs in ECE253. Download `mux.sv` from Quercus and copy (or type) the contents into a file called `part1.sv` on a UG machine. Next, follow the instructions in the submission instructions PDF to submit just `part1.sv` for Lab 1. You can submit the file using the `/cad2/ece253f/public/submit <lab#> <part1 file name>` command; e.g: `cad2/ece253f/public/submit 1 part1.sv`. Afterwards, you can check that the file was successfully submitted using the `/cad2/ece253f/public/check submission <lab#>` command.

As explained in the submission instructions document, you can run the provided tester to make sure your file can be marked correctly. Figure 4 shows the output you will see when you run the tester for `part1.sv`.

When you have finished testing your code, you can submit it for marking. Figure 5 shows the output you will see when you run the marker for `part1.sv`.



```
ug166:~/ece253/lab1% ls
part1.sv
ug166:~/ece253/lab1% /cad2/ece253f/public/1/tester
#######################Part1##########################
Start time: 16:41:05 on Sep 12,2022
vlog part1.sv
Model Technology ModelSim - Intel FPGA Edition vlog 10.6d Compiler 2018.02 Feb 24 2018
-- Compiling module mux
-- Compiling module mux2to1

Top level modules:
        mux
End time: 16:41:05 on Sep 12,2022, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
Reading pref.tcl

# 10.6d

# vsim -c -do "/cad2/ece253f/public/1/test/run.do" work.part1_tb
# Start time: 16:41:06 on Sep 12,2022
# Loading sv_std.std
# Loading work.part1_tb
# Loading work.mux
# Loading work.mux2to1
# do /cad2/ece253f/public/1/test/run.do
# Check mux
# input = 0000000000 output = zzzzzzzzz0 golden_output = 0000 PASSED
# input = 0000000001 output = zzzzzzzzz1 golden_output = 0001 PASSED
# input = 1000000000 output = zzzzzzzzz0 golden_output = 0000 PASSED
# input = 1000000010 output = zzzzzzzzz1 golden_output = 0001 PASSED
# End time: 16:41:06 on Sep 12,2022, Elapsed time: 0:00:00
# Errors: 0, Warnings: 7
######################SUMMARY of part1#########################
Number of Errors by compiling the verilog code: 0
Number of Errors by running the simulation: 0
Number of PASSED: 4
Number of FAILED: 0
part1 is done!
```

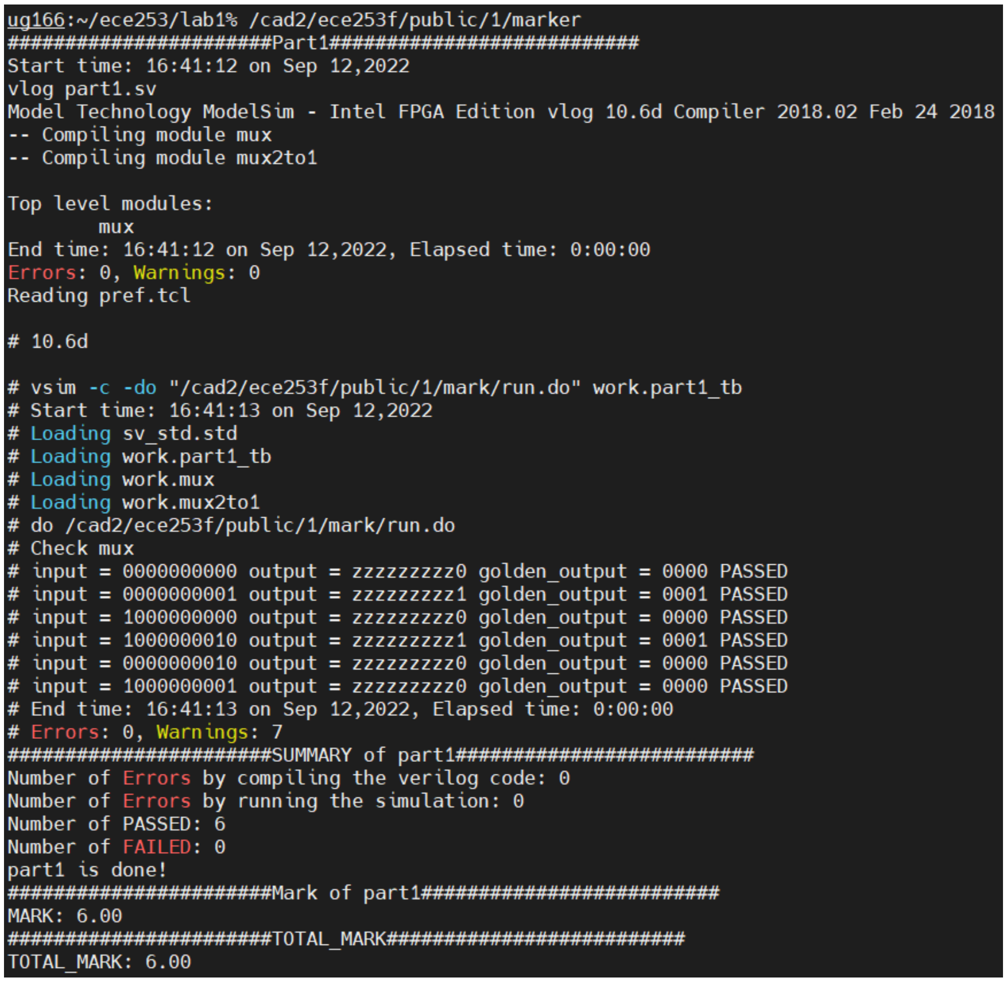
Figure 4: Output from tester for part1.sv

For now, you will not be able to understand what you are seeing as output. That's OK. You will learn about testing your circuits in the next lab. This submission is to make sure you can: 1) login to the UG machines and 2) submit your code for marking and 3) run the tester and auto-marker for your code.

Notice that the marker runs more test cases than the tester. You should keep this in mind for all future labs! Running the tester alone is NOT sufficient to make sure your code works fully. We will provide more information about the tester and marker, starting with Lab 2.

7 Start to Learn Quartus and ModelSim

Starting with Lab 2, you will be using the ModelSim tool to test your designs. You can also (optionally) use the Quartus Prime tool to build and run your programs on the FPGA (either the DE1-SoC or the DE10-Lite). You may find it easier to install these tools on your



```
ug166:~/ece253/lab1% /cad2/ece253f/public/1/marker
#######################Part1##########################
Start time: 16:41:12 on Sep 12,2022
vlog part1.sv
Model Technology ModelSim - Intel FPGA Edition vlog 10.6d Compiler 2018.02 Feb 24 2018
-- Compiling module mux
-- Compiling module mux2to1

Top level modules:
        mux
End time: 16:41:12 on Sep 12,2022, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
Reading pref.tcl

# 10.6d

# vsim -c -do "/cad2/ece253f/public/1/mark/run.do" work.part1_tb
# Start time: 16:41:13 on Sep 12,2022
# Loading sv_std.std
# Loading work.part1_tb
# Loading work.mux
# Loading work.mux2to1
# do /cad2/ece253f/public/1/mark/run.do
# Check mux
# input = 0000000000 output = zzzzzzzzz0 golden_output = 0000 PASSED
# input = 0000000001 output = zzzzzzzzz1 golden_output = 0001 PASSED
# input = 1000000000 output = zzzzzzzzz0 golden_output = 0000 PASSED
# input = 1000000010 output = zzzzzzzzz1 golden_output = 0001 PASSED
# input = 0000000010 output = zzzzzzzzz0 golden_output = 0000 PASSED
# input = 1000000001 output = zzzzzzzzz0 golden_output = 0000 PASSED
# End time: 16:41:13 on Sep 12,2022, Elapsed time: 0:00:00
# Errors: 0, Warnings: 7
#######################SUMMARY of part1#########################
Number of Errors by compiling the verilog code: 0
Number of Errors by running the simulation: 0
Number of PASSED: 6
Number of FAILED: 0
part1 is done!
#######################Mark of part1#########################
MARK: 6.00
#######################TOTAL_MARK#########################
TOTAL_MARK: 6.00
```

Figure 5: Output from marker for part1.sv

own computer, especially if you have ordered an FPGA. To get started with this, read the *Guide to Tools for ECE253* on Quercus.

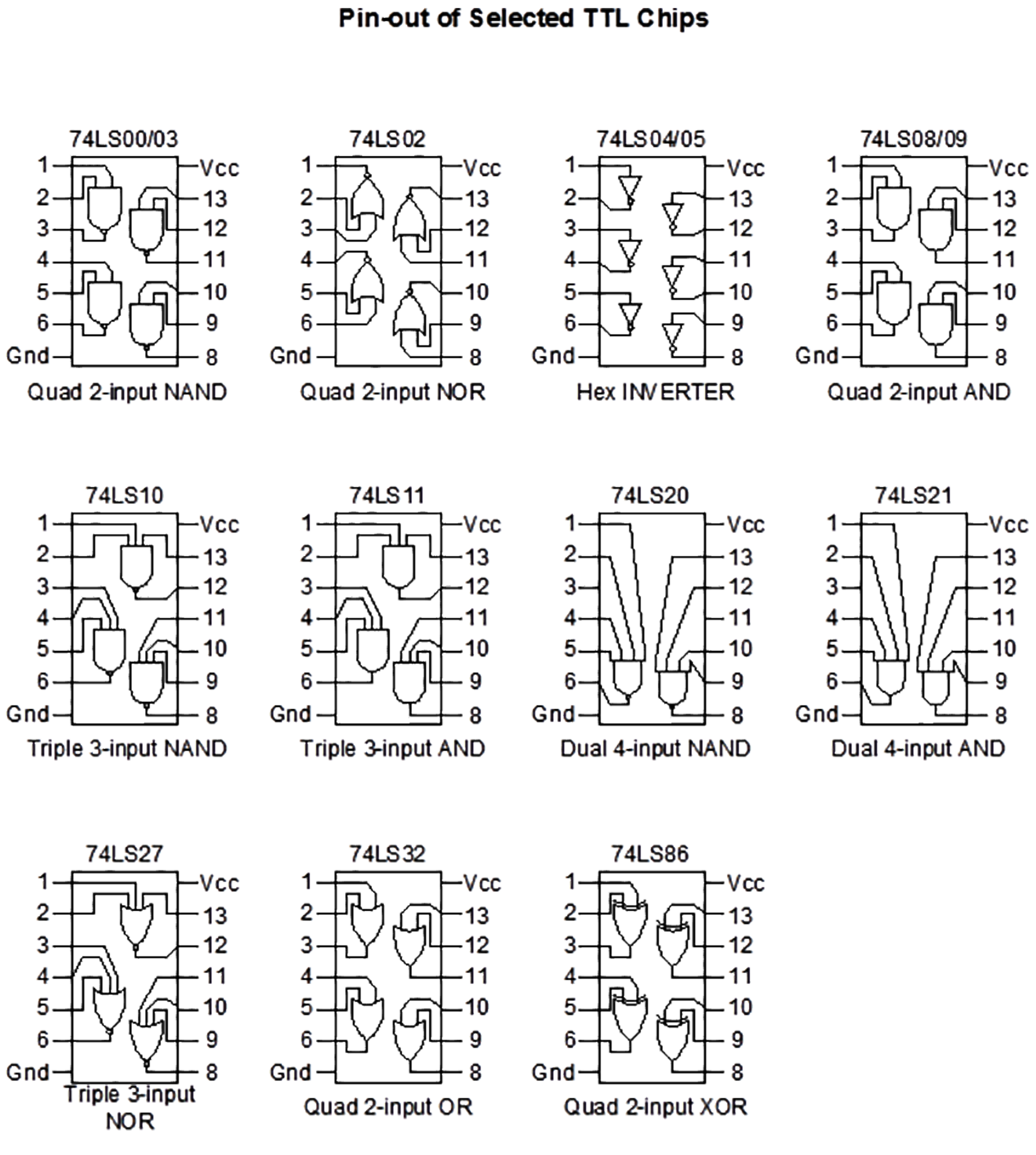


Figure 6: Pin-Out Information for 7400-series Chips

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 7: Digital Board Header Pin Assignment