DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF BRITISH COLUMBIA
CPEN 211 – Introduction to Microcomputers
Lab 2: Building Circuits Using 7400 Series Discrete Logic
Week of September 16 - 20
NOTE: This lab MUST be done individually (or a grade of zero will be assigned)

1. Introduction

In Lab 2 you get experience building digital circuits out of discrete logic components. Even if you have never built anything out of electronic components you will probably have a lot of fun in this lab. While most circuits these days are integrated into a single "chip" understanding the digital design concepts used to build these "chips" is best learned by starting out building small circuits out of discrete components as done in this lab. Completing this lab will help you better understand Verilog and thereby do better in the remainder of CPEN 211.

This lab requires no prior experience building circuits. Before the lab you should at a minimum read this entire handout, familiarize yourself with the necessary components from your tools and parts kit. You should also think through how you will build the circuit for the exercise in Section 2.2. *This lab is out of 10 marks but there are two bonus exercises on the last two pages. Each bonus is worth 1 mark.* You can do both bonus exercises for up to two marks extra (note that only some labs will have a bonus). If you are comfortable doing so, you may work through the lab at home but you should ensure you wear safety glasses. ***The equipment you need for Lab 2 can be found in your 2nd Year Tools and Parts Kit. See complete list of required parts for Lab 2 in Section 3 at the end of Page 11.***

2. Lab Procedure

This section walks you through building a 2-input OR gate out of two NOT-gates and a NAND gate. By following through this part you learn the practical skills required to complete the main exercise. This tutorial is broken into steps. Each step builds upon the prior step.

Always wear safety glasses while working with the equipment for Lab #2. Never leave a battery connected to your circuit while it is unattended. If you find the lab intimidating it is recommended you not connect power for Step 1 or 2 before a TA looks at your circuit. However, if you want to complete the lab at home be 100% sure you have hooked up each component correctly. If not there is a chance some part will drain a large current, heat up and be damaged.

2.1. Tutorial [9 marks total]: Building an OR Gate out of NAND and NOT gates

Step 1 [5 marks]: Setting up your power regulator; a simple test circuit

First, we set up the power supply, and then build a very simple test circuit you can use to verify the light-emitting diodes (LEDs) work (they should, but it is nice to be sure).

The circuit we will build is illustrated in schematic form in Figure 1. To understand this circuit, consider that the battery provided in the lab kits is rated at nine volts. The actual voltage varies slowly as the battery drains. On the other hand, the discrete logic chips we will use in this lab require a steady supply of five volts. We use the LM7805 chip to step down the voltage from 9V to 5V. In our circuits 5V will represent logic value '1' and 0V will represent logic value '0'. We connect the 5V output of the regulator to one side of a switch. We connect the other side of the switch to the positive (or "anode") side of the LED. The negative (or "cathode") side of the LED is connected in series with a 1 kohm resistor. The other side of the resistor is connected to ground. The resistor limits current going through the LED. Without this resistor the diode will "burn out" *very* quickly. Always ensure you have a series resistor when using an LED.

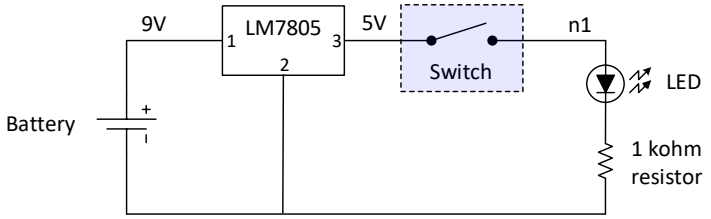


Figure 1: Power supply and LED test circuit - schematic diagram

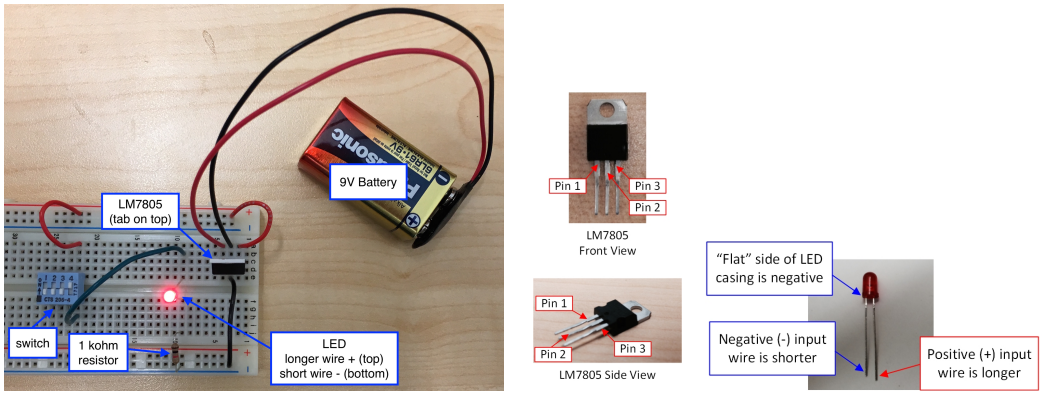


Figure 2: Power supply and LED test circuit – impl. (left); pinout for 7805 (middle), LED (right)

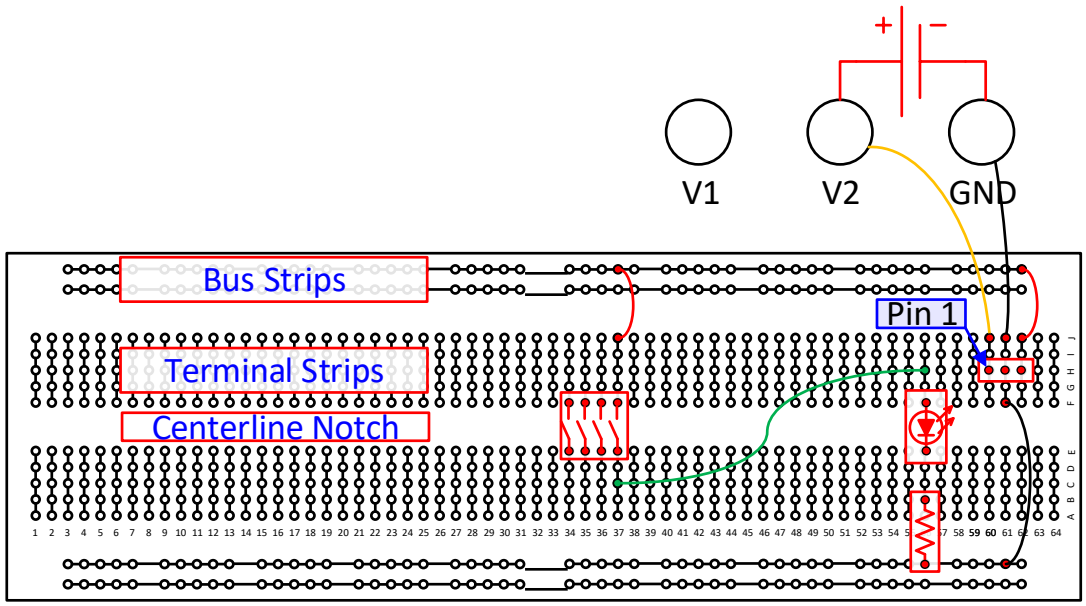


Figure 3: Breadboard connection diagram. Note: Your breadboard may not have V1, V2 and GND terminals. In that case, see how the battery is connected in Figure 2.

We can implement the circuit on a breadboard as illustrated in Figure 2. The different colored wires in the figure have identical electrical properties. Note the power regulator is on the right side of the breadboard versus the left side in the schematic. Figure 2 (middle) illustrates how pins are numbered on the power

regulator. To better understand how the circuit in Figure 1 is implemented on the breadboard in Figure 2, Figure 3 illustrates the electrical connections that run inside the breadboard between the holes in its surface. ***NOTE: The breadboards we will be providing have slightly different numbering than that shown in this figure. The main thing to observe in*** Figure 3 ***is how components are connected.*** Figure 3 shows one of many ways you could build the circuit specified in Figure 1 using the breadboard. The columns of five holes connected with lines are called *terminal strips*. In Figure 3 there are two sets of terminal strips separated by a *centerline notch*. At the top and bottom are long parallel sets of holes used for power and ground called *bus strips*. The terminal strips are arranged so that five adjacent holes in each vertical column in Figure 3 are connected electrically inside the breadboard. The vertical lines between the circles in Figure 3 represented this electrical connection. There are two parallel terminal strips and each vertical column in Figure 3 is given a number between 1 and 64 (the MB-104 breadboards have columns between 1 and 63). Similarly the horizontal rows of holes in Figure 3 are given labels A-E for the bottom terminal strip and F-J for the top terminal strip.

As illustrated in Figure 3 the LM7805 power regulator is inserted in row H with pins in columns 60, 61 and 62. The switch is in a dual-inline-package (DIP) that straddles the centerline notch. One row of pins in the switch is inserted into row E, columns 34-37. The other row of pins is inserted into row F, column 34-37. **The LED is inserted with longer wire (positive) side in row F, column 56 and the other side (negative) in row E, column 56.** The 1 kohm resistor has rings colored brown-black-red-gold (if interested, the color coding scheme is here: http://en.wikipedia.org/wiki/Electronic_color_code#Resistor_color-coding). The resistor has one end inserted in the bottom bus strip in column 56 and the other end inserted in row B of column 56. The wires are inserted to connect the components as illustrated in both Figure 3 and Figure 2. The switches either connect or disconnect the wires on opposite sides of the DIP package. They are connected when the switch is towards the arrow labeled "ON".

Wire up the circuit as illustrated in Figure 3 and Figure 2 and verify that the LED light will turn on and off when you change the switch position. Below we provide some practical tips for constructing the circuit followed by some debugging tips in case your circuit does not work.

Practical tips: The short wires in Figure 2 are 5 cm and medium wires are about 8 cm long. The green wire is 11 cm. The color of the wire does not change the electrical properties. However, you will want to use the colors strategically so you can easily remember what different parts of your circuit do. I suggest using red (or orange) for 5V and black for ground. To remove insulation place the wire inside the center of the wire stripper Figure 4 (left). It is easier to strip wire if you use the needle-nose pliers to hold the wire as illustrated in Figure 4 (middle). You will want to strip about 0.5 cm of insulation off the end of the wire, as shown in Figure 4 (right). Figure 5 (left) illustrates how to cut wire using the innermost part of the wire cutter/stripper and using the needle-nose pliers to keep the wire from being pushed out of the wire cutter. Using the needle-nose pliers can be helpful when inserting wire into the breadboard Figure 5 (center). For the wire ends connected to the power terminals (see Figure 2), strip about 1 cm of insulation. Avoid

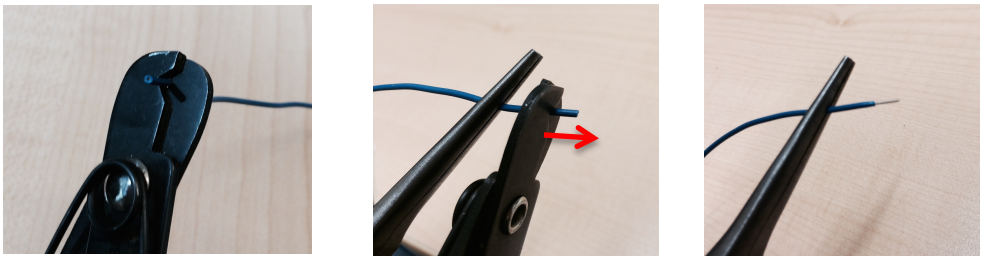


Figure 4: Practical tip – stripping insulation with help of needle-nose pliers

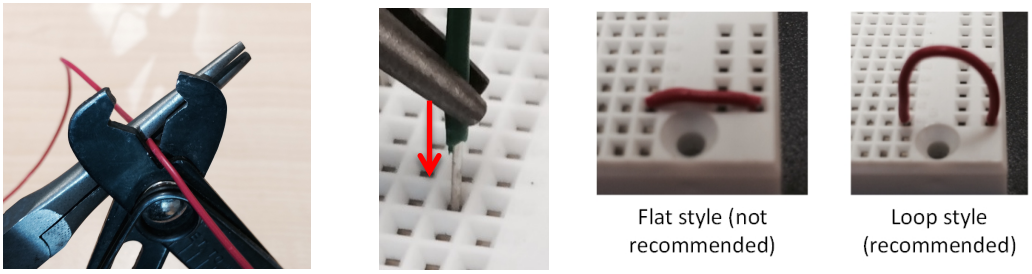
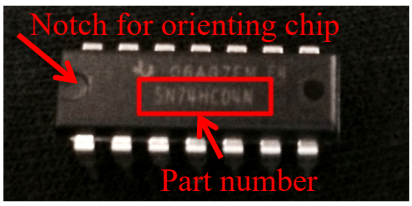
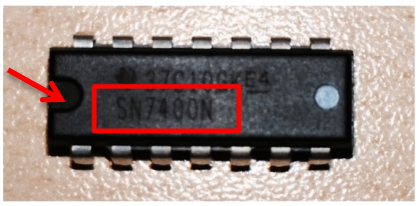


Figure 5: Cutting wire (left); inserting wire w/ needle-nose pliers (center); wiring styles (right)

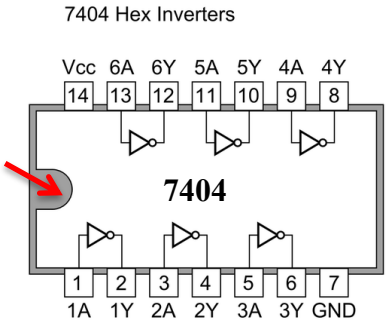


(a) HEX Inverter chip (7404)



(b) NAND (7400)

Figure 6: Identifying chip type and finding orientation notch



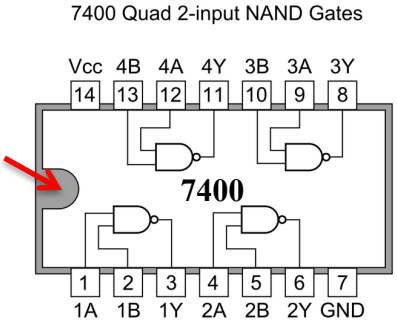


Figure 7: Pinout diagram for 7404 and 7400 TTL chips

making sharp bends in the wire – these can cause the end of the wire to break off inside the breadboard. Instead of laying wires flat on the board, make wires a bit longer and arc them over other components Figure 5 (right). ***Connect LEDs in series with a 1 kohm resistor (otherwise, the LED will "burn out").***

Debugging Tips: If the LED will not light up be sure you connected one side of the switch to 5V. Check if any wires are loose. Next, check you did not reverse the LED pins. Reversing the pins on the LED will cause no current to flow and no light to be emitted because current can only flow one way through a diode (as you will learn in EECE 251). Check you have connected the power regulator correctly. Make sure your wires are inserted into the breadboard correctly and that no wires have inadvertently broken off after being inserted into the breadboard (this can potentially happen if you accidentally nick the metal part of the wire when stripping off the insulation). Be sure the thumbscrews on the power terminals are tight so the wires there are not loose. Remember to connect the 9V battery too. Finally, test the battery (the TA can show you how to do this using the multimeter in the lab) but note that the ones we handed out are new.

Step 2 [2 marks]: Adding a NOT gate

First, some background on five chips we give you. There are two important pieces of information you should look for when handling the chips. First, each chip has a part number printed on top. See examples in Figure 6. Your kit should contain three 7400 and two 7404 chips. Note that the exact part number on your chips may include some additional letters between some of the numbers (e.g., "74HC00AP" instead of 7400, "SN74HC04N" instead of 7404, and others; ***each kit may have slightly different codes***). For our purposes, we don't really care too much about what these letters mean. For those who are curious, "H" means high-speed, "C" means CMOS (more details: http://en.wikipedia.org/wiki/7400_series). However, all the chips we give you should use 5V supply voltage, which is the only "feature" relevant to this lab (there are some 7400 series chips that use 3.3V). A notch on the top of the chip conveys the second important piece of information. The notch identifies the orientation of the chip and hence enables us to identify the pins to match them up to a *pinout* diagram of the chip.

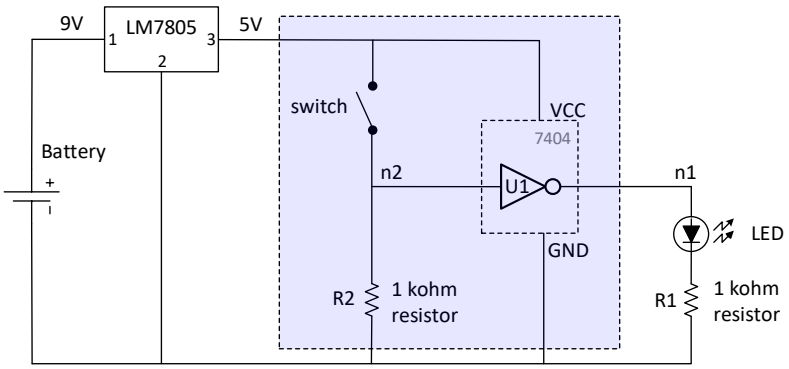


Figure 8: Adding a NOT gate -- schematic

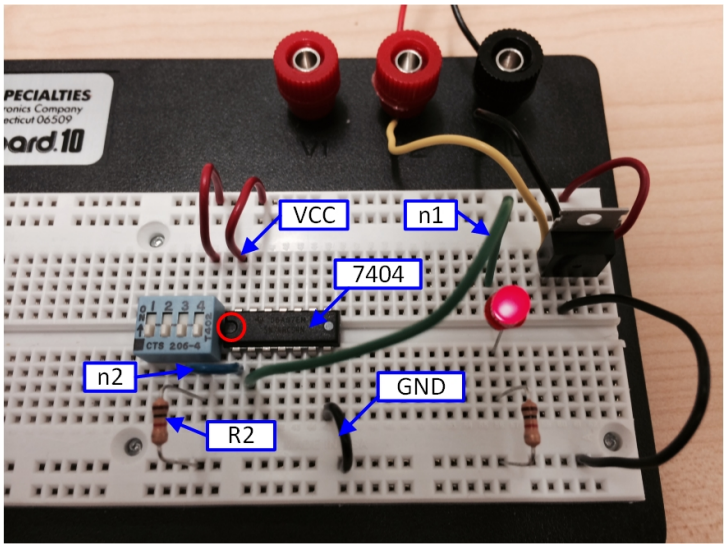


Figure 9: Adding a NOT gate -- implementation

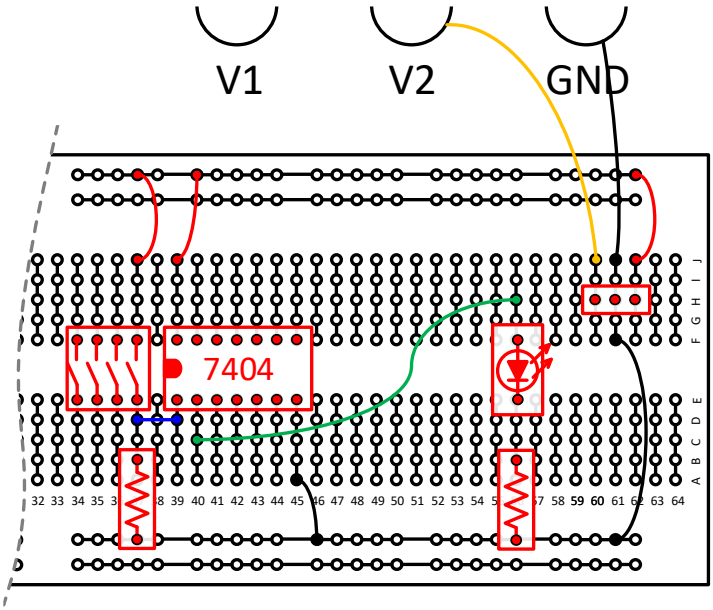


Figure 10: Adding a NOT gate -- connection diagram

While the pins are labeled with numbers in Figure 7, on the actual chips in Figure 6 there are no pin numbers. To identify which pin connects to which gate terminal orient the chip by identifying the notch highlighted by red arrows in Figure 6. Then count off the pins in a counter-clockwise direction from the bottom left.

Okay, let's add that NOT gate! The new schematic is in Figure 8. We replace the part highlighted in blue in Figure 1 with the part highlighted in blue in Figure 8. We add a second resistor, R2, and a 7404 chip including a NOT gate, U1. We connect the 7404 power supply input (VCC) to the 5V output of the power regulator and the ground input (GND) to the negative terminal of the battery (which is 0V or "ground"). Wire "n1" now connects the LED to the output of the NOT gate instead of directly to the switch. The input to the NOT gate, wire "n2", should be at either 0V or 5V to represent inputs of logic value '0' or '1'.

You might wonder, "how can we make n2 take on these two voltages with a simple open/closed

switch?" In some sense the ideal solution to generate a logic '1' or '0' input to our circuit would be to use a "two-throw" switch that connects an output terminal to either a 5V terminal or a ground terminal depending upon the switch position. Such switches need three pins per switch, which complicates using them with a breadboard. A common alternative, which we have employed in Figure 8, is to connect one side of a "single throw" switch (such as the ones we give you) to the 5V supply and the other side to a resistor that is in turn connected to ground. When the switch is in the "ON" position (closed/connected), some current will flow through the switch and resistor and the output (e.g., "n2") will be at 5V. When the switch is in the "OFF" position (open), no current will flow through the resistor and the output (e.g., "n2") will be at 0V since the voltage across a resistor is zero when no current flows through it.

Wire up the circuit in Figure 8 as illustrated in Figure 9 and Figure 10. You should find that now the LED turns on when the switch is in the OFF position and turns off when the switch is in the ON position. Now, you may have been wondering, or you *should* have been wondering, "how did we know which pins on the chip to connect wires too?" The answer is we very carefully looked at the chip pinout diagrams in Figure 7 and using the notch on the top of the actual chip to orient the chip, we determined which pin is which. You will notice from Figure 7 that each of the chips has either four NAND gates (for a single 7400) or six NOT gates (for a 7404). When using the chips for the exercises in this lab and Lab 3 you will need to regularly consult Figure 7 so you should be very sure you understand why each wire is connected to the chip the way it is in Figure 9 and Figure 10.

Some practical tips: It can be challenging to insert the 7404 chip into the breadboard with pins in rows E and F. One solution is to instead use rows E and G. Another is to use your needle-nose pliers to grab one row of chips as shown in Figure 12 and gently bend the entire set of pins *very slightly* inward. Remember to connect power and ground to your logic chips or they will not work. Double-check the orientation of your chips before connecting them up. See the notch highlighted in red in Figure 9 and Figure 10. If you need to remove a chip after inserting it into the breadboard, stick one tine of your needle-nose plier into the centerline notch under the chip so as to pry it loose as illustrated in Figure 11. **Be very careful not to break any pins on the chips. We may not have enough spares to give you a replacement chip.**

Debugging Tips: Verify you have correctly connected power to the 7404. To debug your circuit a long wire connected a spare LED and series resistor can be used as a "logic probe" to test for logic '1' or '0'.

Figure 12: Practical Tip – gently bend pins slightly inward for easier insertion into breadboard

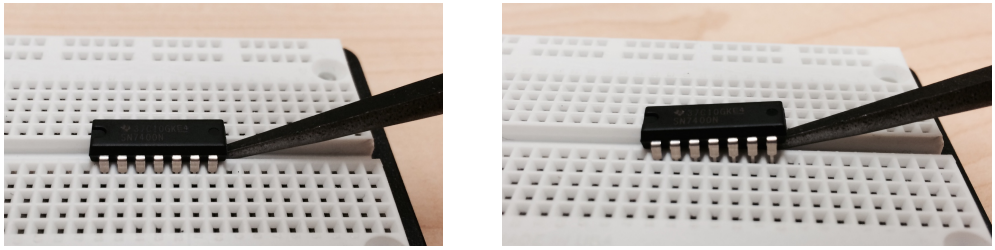


Figure 11: Practical Tip – gently pry chip from board by inserting pliers in centerline notch

Step 3 [2 marks]: Adding a Second NOT gate and a NAND gate

To build interesting circuits, we need to combine multiple logic gates. But, what type of gates do we need for our eventual circuit? It turns out that NAND gates are *universal gates.* What this means is using only a sufficient number of NAND gates we can build *any* digital circuit. If we connect both inputs of a NAND gate together and treat it as a single input we get a NOT gate. A circuit combining another NAND-gate followed by that NOT-gate acts as an AND gate. A circuit consisting of a NAND gate preceded by two NOT gates, one on each input, is an OR gate. An n-input AND gate can be built by combining n-1 two-input AND gates and the same trick works for OR-gates. With inverters (i.e., NOT gates), n-input AND gates and n-input OR gates we can implement any Boolean function by converting the truth table to sum-of-products form.

Since converting a NAND-gate into a NOT-gate wastes two transistors, we instead provide you with both NAND-gates and NOT-gates.

The schematic diagram for our OR-gate is illustrated in Figure 13. The portions highlighted in blue have been added or modified from Figure 8. On the left we add a second logic input using a switch and series resistor. In the middle we add a second inverter. We connect the output of both inverters to the inputs of the NAND gate. Note that this combination of two-NOT gates and an NAND gate together compute the same logic function as an OR-gate. The output of the NAND gate drives the LED.

Wire up the circuit as shown in Figure 14 and Figure 15 and verify it works as an OR gate. Note that Figure 14 only labels the parts of the circuit that have changed. By "verify" I mean you should make sure the output is a '1' (the LED is "on") if either input switch is set to "on" and is only '0' (LED is off) only if both switches are "off". You should be able to explain why the combination of two NOT gates and a NAND gate is an OR gate (your TA may ask). If you cannot, be sure to review the lecture notes on logic gate diagrams from Slide Set #2 before continuing. **You may ask for help understanding everything in this handout up to this paragraph on Piazza. The exercises below you should complete on your own.**

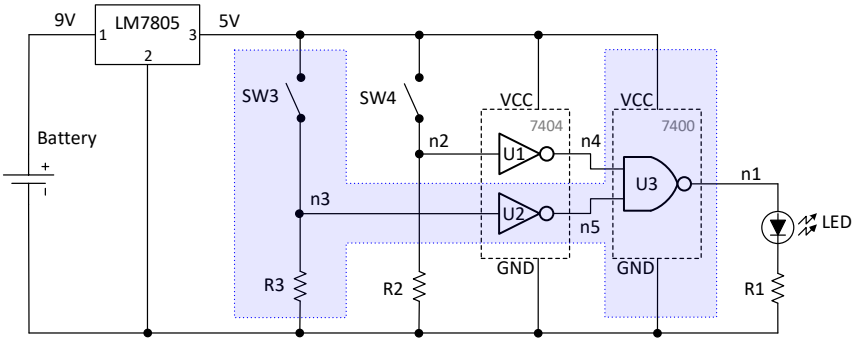


Figure 13: Adding a Second NOT-gate and a NAND-gate -- schematic

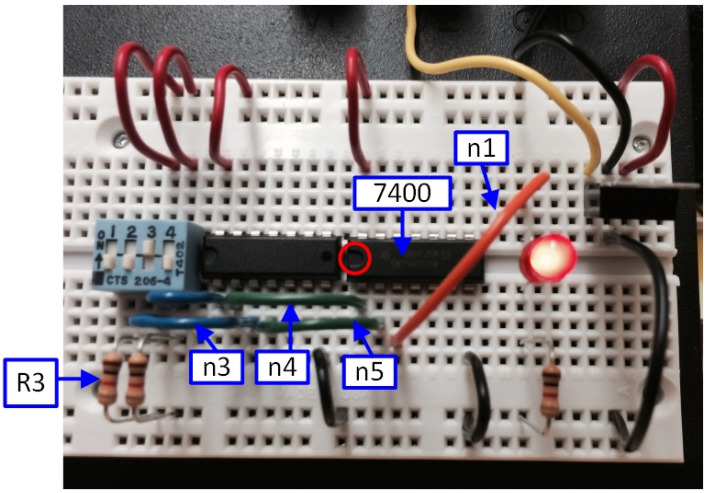


Figure 14: Adding a Second NOT-gate and a NAND-gate -- implementation

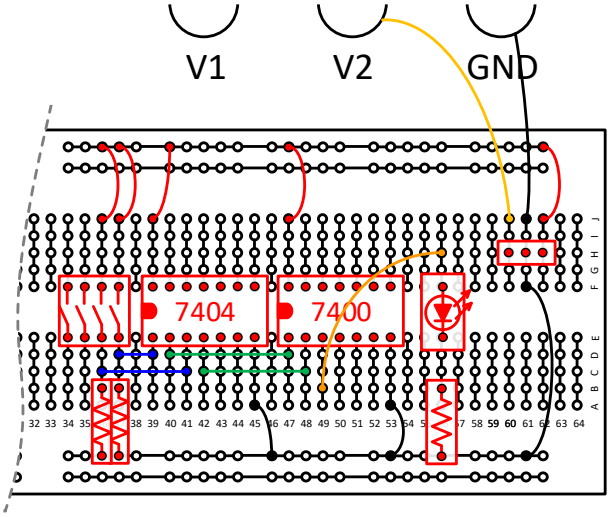


Figure 15: Adding a Second NOT-gate and NAND-gate -- connection diagram

2.2 Exercise [1 mark]: Implementing a more complex Boolean function

Build an *optimized* circuit to implement: $f = (\bar{a} \wedge \bar{b} \wedge \bar{c}) \vee (\bar{a} \wedge b \wedge \bar{c}) \vee (a \wedge \bar{b} \wedge \bar{c}) \vee (a \wedge \bar{b} \wedge c) \vee (a \wedge b \wedge \bar{c})$ To avoid needing to rewire your circuit during your demo, use different gates for this part. These "new" gates can be on the same 7400 and 7404's as the gates used for earlier steps. Also, it is important to first optimize the logic expression by simplifying it using the rules of Boolean algebra, or you will not have enough parts! Note, while it may be obvious how to implement this Boolean function using AND, OR and NOT gates it is less obvious how to do this using NAND gates and NOT gates. So, you will need to think a little bit about how to do this. The key material is in Slide Set 2 were we talked about logic gate diagrams. You will also want to review the material on Boolean algebra and specifically De Morgan's Law from Slide Set 1. Note that it is possible to simplify the given expression using the Boolean logic properties from Slide Set #1. Doing so will reduce the number of gates you need to connect. Wire up the circuit and connect it to the LEDs and switches to demonstrate it to your TA. ***Do NOT implement a logical OR by directly connecting the output of two gates (or by connecting one of the slide switches directly to the output of a gate) as this burns a significant amount of power and may damage the chips.***

Practical Tips

1. After drawing your schematic diagram of your circuit but before doing any wiring, label each NAND and NOT gate in your schematic as U1, U2, U3, etc... Then, draw a separate rough sketch of your breadboard showing the 7404 and 7400 chips and indicating inside of them where each of those U1, U2, U3, etc… chips will be on your breadboard. Do not bother trying to draw the wires on this diagram because it will become too messy. For example, for the schematic in Figure 13 you might draw a simple diagram like that shown in Figure 16 which corresponds to the layout used in Figure 14 and Figure 15. Such a "gate placement diagram" is helpful when your circuit is half way finished and something interrupts you. Without such a diagram you may very easily forget where you put each gate in your circuit. Synthesis tools follow a similar two-step procedure known as "place and route" when building *very-large scale integrated* (VLSI) chips. So does Quartus when mapping your VHDL down to logic elements in the FPGA. This tip will be even more important in Lab 3.
2. As you add wires to the breadboard add a small checkmark beside the corresponding wire in your schematic so you can easily keep track of which wires are done and which still need to be added.
3. Use different colored wires to make it easier to identify different parts of your circuit later (for example, when the TA asks you to explain how your circuit works).

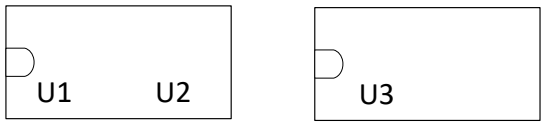


Figure 16: Practical Tip -- Decide placement of *all* gates before connecting *any* wiring

Debugging Tips: An excellent reference on debugging is Agans (See - UBC eLink: http://goo.gl/MgvFRc)

2.3 Bonus Exercise #1 [1 marks]: Build a "Flip-Flop"

To complete this part you will need 8 NAND gates and three NOT gates. In lectures we will soon be learning more about sequential logic circuits. For this part of the lab, you will build a simple memory element called a flip-flop, which is a key part of all sequential logic circuits. We will briefly talk about the internal operation of a flip-flop in lecture, but there is no substitute for having built one. Drive the inputs to the circuit in Exercise 2 using the switches from Exercise 1. You must show both designs for your demo.

The first step is to build what is known as an RS-latch and pictured in Figure 17. The operation of this circuit can be understood as follows. When **S** and **R** are both '1', the circuit is said to be quiescent. In this state, if output **a** is '1', then input **A** is '1' and since **R** was assumed to be '1', then output **b** is '0'. If output **b** is '0' then input **B** is '0' and consequently output **a** is '1'. So, the overall combination of two NAND gates is in a stable state. If instead we had assumed **a** was '0', we would find that **b** must be '1' and again the combination of two NAND gates does not change value. So, we can see that with S and R values of '1' this circuit "remembers" what state it was in. (Note this version of an RS-latch is slightly different from the one in the textbook. The version in the textbook uses two NOR gates and it is quiescent when ***r*** and ***s*** inputs are both '0').

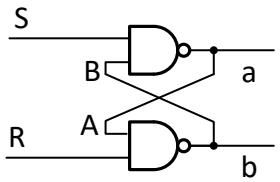


Figure 17: R-S Latch

If we instead we have **S**='0', and **R**='1', then **a** must be '1' and so **b** must be '0'. If we have **S**='1' and **R**='0' then **b** must be '1' and so **a** must be '0'. Hence, we can set and reset the value of **a** by lowering **S** or **R** while keeping the other input at '1'. Build the circuit in Figure 17 on your breadboard and verify it works as described.

A more useful circuit is the gated D-latch pictured in Figure 18. It consists of an RS-latch preceded by a pair of NAND gates. When write-enable (**WE**) is '1' the value of **D** is passed through to the output **Q**. When **WE** is '0' the output **Q** holds its current value. When Quartus says it is "inferring a latch" (e.g., Lab

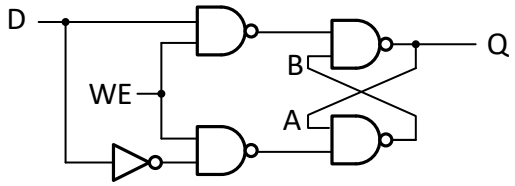


Figure 18: Gated D Latch

#1) that means it is generating a circuit like that shown in Figure 18. Usually, Quartus does this when we have incorrectly specified a Verilog "`always`" statement that was really meant to describe combinational logic—logic without any feedback loops like the feedback loops in the RS-latch. Starting from your R-S Latch, build Figure 18 on your breadboard using 7400 and 7404 chips.

Finally, we can build a circuit where the output only changes when the input is changing from logic value '0' to logic value '1' using the circuit illustrated in Figure 19. This circuit is known as a Master-Slave Flip-Flop and we will talk more about it in class. When **CLK** is '0' M tracks **D**. When **CLK** change to '1' **Q** takes the value of **M**, effectively sampling the value of **D** on the "rising edge" of **CLK**. Implement the master-slave flip flop circuit exactly as illustrated in Figure 19 while connecting **D** and **CLK** inputs to two switch-resistor pairs and the **M** and **Q** outputs to and LED with a current limiting resistor. Experiment with this circuit. **Q** should change to the value of **D** only when **CLK** changes from '0' to '1'. Challenge: See if you can figure out why.

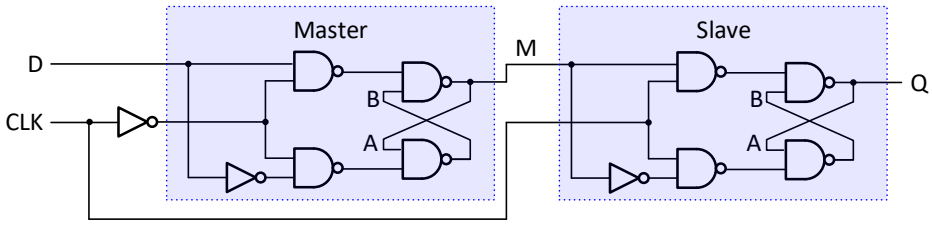


Figure 19: Master-Slave Flip-Flop

Your group mark for this section will be computed as follows:

[0.5 marks] If you get an RS latch to work on your breadboard.
[0.5 marks] If you extend the RS latch to work as a gated D latch on your breadboard AND you combine two gated D latches with a NOT gate into a working master-slave flip-flop using the included push buttons AND you can explain why **Q** changes when **CLK** goes from '0' to '1' for the master-slave flip-flop.

NOTE: You will need to think carefully how to demonstrate to the TA that your circuit for this part works.

2.4 Bonus Exercise #2 [1 Mark]:

Connect exactly three NOT gates in series so the output of one connects to the input of the next and back again to the first as illustrated in Figure 20. Probe the output of any one of the NOT gates with the oscilloscope above your workbench in the lab. **NOTE:** you will need to figure out how the oscilloscopes work by yourself **without the TA's help**—the oscilloscope brand and model in MCLD 112 is Tektronix TDS 2012 "Two Channel Digital Storage Oscilloscope". ***If you can correctly show the TA the output of one of the NOT gates on the oscilloscope and explain why it looks the way it does, you get one bonus mark.***

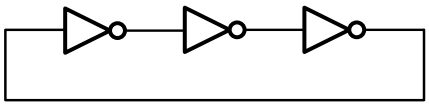


Figure 20: Bonus circuit (challenge: what does it do?)

3. Required tools and parts for Lab 2 (found in your 2nd year tools and parts kit).

Safety glasses!
1 × breadboard
1 × switch block
3 × 7400 NAND (5V)
2 × 7404 NOT (5V)
1 × LM7805C
2 × pushbutton switch
4 × ~1 meter length of wire (various colors)
1 × 9V battery clip
1 × wire cutter & stripper
1 × needle nose pliers
8 × 1 Kohm 1/4 watt 5% resistors
4 × red light emitting diodes
9V battery