Digital Lab 2

Preparations:

- Read this entire lab write up before coming to lab.
- From “Exploring Raspberry Pi, read Chapter 4 the section on Logic Gates.
- From "Practical Electronics", read Chapter 12, especially section 2 (Logic Gates) and 6 (Flip-Flops).

Goals:

- Learn about logic gates and flip-flops.
- Learn about the logic analyzer.
- Learn to connect logic to the Raspberry Pi.

Expectations:

- Keep good notes in your lab book. It seems we cannot say this enough.
- Accurately execute the tasks in this lab.
- Improve breadboard skills.
- Improve skills interfacing the Raspberry Pi.

Introduction

We got a taste of what we can do with a Raspberry Pi in the first lab, but clearly there is a lot more. In this lab we will start exploring the possibilities a bit further.

Logic and RPi GPIO

Introduction

All digital electronics are formed with logic gates. The three most basic ones are AND, OR, NOT, and to some extend the XOR. The NOT gate is also called an inverter, since that is what it does, invert the signal. Other very commonly used logic gates are combinations of these 4: NAND (NOT AND) and NOR (NOT OR). The XNOR (NOT XOR) is pretty rare, and would usually be build from an XOR followed by an inverter.

For this lab we have the SN74HC00N, quad NAND gate, which means it has 4 NAND gates, the SN74HC02N, quad NOR gate, the SN74HC86N, quad XOR gate, and the SN74HC04N, hex inverter (i.e. 6 NOT gates). Look up the datasheets for how to connect them.

CMOS, TTL and Powering Logic Chips.

Note that we are using modern CMOS type gates in this class, which have a Vcc (the input voltage for the chip) between 2 and 5 Volts. If you are planning to use the gates with the Raspberry Pi, **you must use 3.3 V for all your logic chips.** [1]

[1] Well, OK, refinement, you must use 3.3V for the logic chip if you plan to read an output from the logic chip with an input GPIO on the RPi. It turns out that the other way around, though not quite “in spec” (within the stated specifications on the data sheet), you can steer the **input** of a logic chip running at Vcc=5V with the 3.3V **output** of the RPi, and it will actually work. This is a special topic, called “level shifting”. For now, please, power your logic chips at 3.3V.

BEFORE YOU PROCEED

Can't put this bold enough!!! **CHECK** with your volt meter to make sure that you have set the power supply of your AD **to 3.3V**. Supply the power for you chips from this 3.3V.

Once you have done this. Quickly check to make sure you understand how a NAND gate works, how a NOR gate works, how the XOR works, and how to use the inverters.

A quick and robust way to check these circuits is to use the Static I/O, and later if you wish the Logic Analyzer, on your AD.

First test the basic functioning:

You want to do this part fairly quickly and not dwell too long on it.

Note: for any of these logic chips (or any other chip) to function, it needs some power. You must connect the Vcc pin to 3.3V and the GND pin to ground. If you fail to do this, your circuit will likely exhibit strange behavior, or not work.

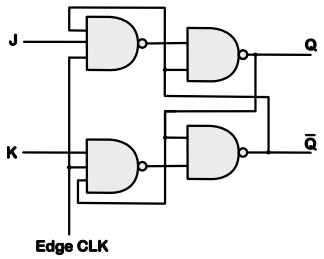
On the AD2, open the Static I/O screen. You see 16 channels labelled from 0 to 15, which are all setup for input, or "LED". Note that some may actually be red, which means 1 or "on". This is because these inputs are CMOS, and so sensitive that a bit of stray charge will turn them on. If you grab some of the input wires, they are likely to blink rapidly. This is your first lesson on working with CMOS digital circuits. All inputs need to be either tied to 0 or 1, otherwise they may oscillate and cause noise in your circuit.

- Convert two of the inputs to outputs (click on the number). Choose switch or button that has a 0 and 1 state. The "Z" means "tri-state", or "open", which is not what you want here.
  - With a voltmeter, check that for "0" the output is close to 0V and for "1" it is close to 3.3V.
  - Connect this button to one of the AD digital inputs, shown as a virual LED, and check that you can toggle the LED.
- Now use the inverter (NOT) chip to invert the signal from the switch, and test it with the "LED", or you can also use a real LED circuit like you build before for the RPi output.
- Now test the functionality of the NAND, NOR and XOR gates, using two buttons on the inputs and an LED on the output. Check these results against the expected results from a logic table.
- **Extra Task:** If you feel inclined, you can also try the Logic Analyzer and Pattern Generator. The first is a "scope" for digital signals, it shows you how a digital signal changed over time. The second generates digital pulse trains of various types. You use these two to get detailed tests of circuits, and they are very handy when you are debugging more complicated circuitry. If you don't get to this, you will learn about it later.

Flip-Flops

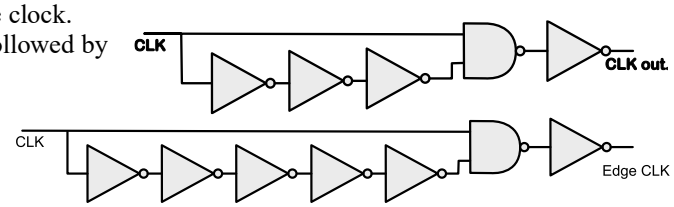
In lecture we talked about how you can make a "flip-flop" using basic gates. This is good practice for building, debugging and understanding digital circuits.

TASKS:



- Use the NAND gates to make an R-S Flip Flop, and test the circuit and the behavior of the circuit.
- Add to your NAND gate R-S Flip Flop to turn it into a D Flip-Flip. Again, test it and test the behavior.
- We do not have 3 input NAND gates, so you cannot make a J-K Flip-Flop in one step. What you *can* do (pick your favorite ones to do, if you have time):
  - You can create a 3 input NAND with the parts you have, though you will need to use another NAND chip to complete a J-K Flip-Flop.
  - Create a J-K Flip-Flop that has the J and K input "missing", which is the equivalent of having those two inputs set to 1. (A 3 input NAND, with one input set to 1, is just a 2 input NAND). If you make this circuit, you can check that is behaves like a clock divider, one pulse out for every 2 CLK pulses in. This is the first stage of a counter circuit.

- **NOTE:** For the J-K Flip-Flop, the **clock** needs to be an edge clock. You make an edge clock with three inverters, a nand gate, followed by a 4th inverter:
  You probably need to use 5 inverters for the delay stage if 3 is not sufficient to get a good pulse. These inverters are fast.



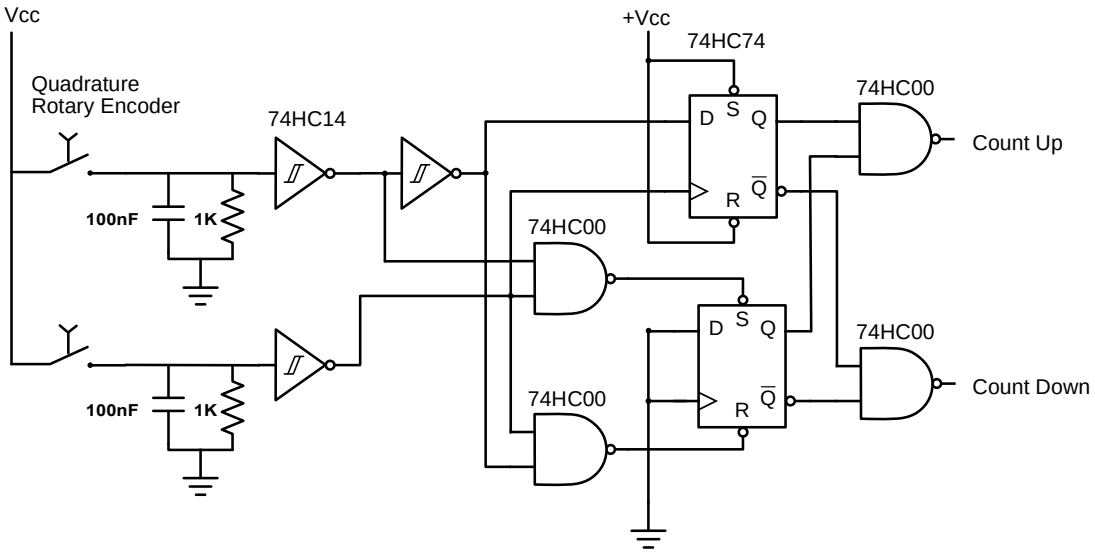
- Extra, extra: If you have two NAND chips, you can make a two stage divider. Now you can count from 0 to 3!
- Extra, extra: With two NAND chips, you can also make a 2 stage shift register.

Optional Challenger: Quadrature Encoded Rotary Encoder

This is a really fun circuit, that you can then use in Lab 6. It is a bit of a challenge to build, but you have all the components in your kit for this. If you make this, put it on a breadboard that you can save for later.

The rotary encoder is a small device with a knob you can turn and push. As you turn it you feel little clicks, and with each click a small switch inside goes open or close. If you push the knob, another switch is activated. These devices are now very common in audio equipment to replace the old-fashioned volume knob. There are only two outputs to readout the rotation, which is encoded in a quadrature signal, see how this works here: http://www.dynapar.com/Technology/Encoder_Basics/Quadrature_Encoder

This is, I think, kind of a fun mini project, that you could build if you have time. In the next digital lab you will learn about counters. It would be nice to have this circuit on a separate breadboard, so you can save it and then use it with your counter in the next digital lab. To be useful the quadrature signal needs to be decoded. You could do this in software, using two input pins on the RPi, but you can also do this nicely in hardware. The needed circuit is simulated here on systemvision: http://sysvis.io/Nd23aw. We should have the needed parts for this system available. Ask if you need some help with this.



The rotary encoders we have in lab are Bourns Rotary Encoder PEC16-4220F-S0024.

Progress Report Expectations:

This again does not need to be a very lengthy progress report.

- You do need to state what you accomplished and any trouble you may have run into.
- You need to have diagrams for each of the Flip-Flop circuits you made in the lab.
  - You can use your own drawing program, or use "Schemeit" from Digikey (a free web app): https://www.digikey.com/schemeit/project.

    - To find the logic parts, click "Schematic Symbols" in the list on the left, and then "Logic" and then "Gates".
- Don't forget that in these shorter progress reports, it is even more important that you state:
    - What you learned.
    - Who you helped.
    - Who helped you.