ELEC 101
April 21, 2005
Prof. Rich Kozick

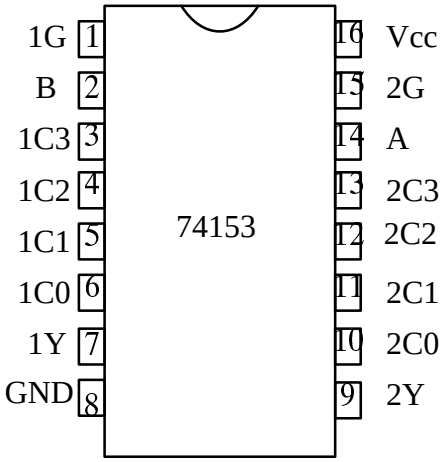
LABORATORY 12
Multiplexers and Flip–Flops

In the first part of this lab, we will learn how to use the 74153 4–line to 1–line multiplexer chip. Besides serving as a data selector, we will see how to use the 74153 to implement the 3–input majority function and a serial–to–parallel converter. In the second part of the lab, we will investigate a simple and widely–used device that acts as a 1–bit memory, the S–R flip–flop. (The S–R flip–flop is the foundation of the D flip–flop.) You can read about both topics in Chapter 12 of Bobrow.

You can use switches and LEDs for your inputs and outputs, respectively.

Multiplexer

The 74153 contains two 4–line to 1–line multiplexers. A multiplexer is a device that connects one of four input lines (C0, C1, C2, C3) to a single output line (Y) as specified by two selector bits (B and A). There is also a "strobe" input (G) that must be set to "0" in order for the multiplexer to function properly. A pin diagram for the 74153 is shown below.



In your lab notebook, record the output Y that you measure for each input in the following truth table. The X means that the input is irrelevant to the output. Verify by inputting both ''0'' and ''1'' for some of the X's. Do you see how this chip functions as a multiplexer/data selector?

Select		Data Inputs				Strobe	Output
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	1	
0	0	0	X	X	X	0	
0	0	1	X	X	X	0	
0	1	X	0	X	X	0	
0	1	X	1	X	X	0	
1	0	X	X	0	X	0	
1	0	X	X	1	X	0	
1	1	X	X	X	0	0	
1	1	X	X	X	1	0	

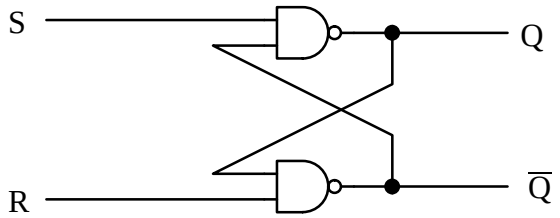
A multiplexer can also be used to implement general logic functions. Configure the 74153 so that it performs the majority function for 3 inputs, and demonstrate. (Refer to Bobrow, Chapter 12, for implementing logic functions with a multiplexer.)

Please design the following with the 74153. You do *not* have to set up these circuits.

1. Design an 8–line to 1–line multiplexer.
2. Design an 8–bit parallel–to–serial converter. Do you need another chip?

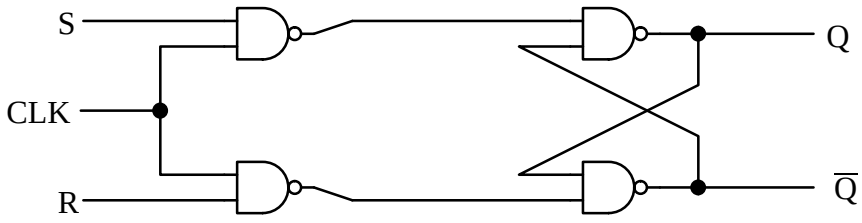
Flip–Flops (If you have time ...)

The figure below shows a NAND gate implementation of a widely–used memory element called an RS flip–flop. The outputs, Q and $\overline{Q}$, are complements of each other during normal operation. Think about how the circuit will work, and write down a truth table in your lab notebook. Take some time to understand the circuit — it is a little different from the logic functions that we have studied so far. Note that the outputs are *fed back* as inputs to the circuit, so past outputs play a role in determining the present output.



After you have analyzed the circuit and developed a truth table, connect the circuit and verify your truth table. You should find that you can ‘‘set’’ Q to ‘‘1’’ by applying a ‘‘0’’ to S. Similarly, you can ‘‘reset’’ Q to ‘‘0’’ by applying a ‘‘0’’ to R. When S and R are both ‘‘1’’, then the previous output is held (i.e., there is no change). What happens when S and R are both ‘‘0’’? Apply each input several times in order to understand how the previous output influences the present output. Do you see how the SR flip–flop serves as a 1–bit memory element?

If you understand the SR flip–flop, consider the ‘‘clocked SR flip–flop’’ shown below. The idea of this circuit is to allow the output states to change *only* when an input clock signal equals ‘‘1’’. The previous output is held when the clock equals ‘‘0’’, regardless of the values of S and R. A clock is used in most digital systems in order to simplify the analysis and design.



Develop a truth table for the clocked SR flip–flop using S, R, and CLK as the input variables. Then connect the circuit, and verify your truth table. When the clock is ‘‘1’’, is there any difference in the way that you set and reset the flip–flop compared with the previous circuit?

How can you use the clocked SR flip–flop to create a D flip–flop? (D = data or delay)