Sequential Logic: Clocks, Registers, etc.

ENEE 245: Digital Circuits and Systems Laboratory
Lab 2

Objectives

The objectives of this laboratory are the following:

- To design various latch and flip-flop circuits
- To test various latch and design circuits
- To measure the non-ideal properties of your circuits and compare the performance of your flip-flop(s) with that of a pre-packaged flip-flop

Latches and flip-flops are the primitive storage devices in sequential circuits. In this laboratory, you will study their functional and temporal behavior and develop some insights about sequential circuit operation in general.

Latches

A latch is an asynchronous digital circuit that has two stable states—0 and 1—that can be used to store state information.

SR Latch

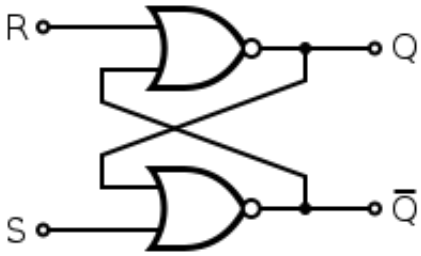


Figure 1 A NOR-based SR Latch

One of the most fundamental latches is the SR latch, where S and R stand for *Set* and *Reset*. Figure 1 shows the logic diagram of an SR latch built using cross-coupled NOR gates. The stored bit is available at the output marked Q; its complement is available at output #Q. While the S and R inputs are both low, feedback maintains the two outputs in a constant state.

When R = 1 and S = 0, output Q will go to 0 regardless of its value before R was set to 1, and that forces #Q to 1 after a brief delay. Thus, R resets the output to 0. When R = 0 and S = 1, #Q becomes 0 and Q becomes 1. Thus, input S sets the latch.

When R = S = 1, both Q and #Q become 0, and are no longer complementary to each other. At this point, if both R and S are simultaneously switched to 0, both outputs will be forced to become 1, which in turn will try to force both outputs to become 0, and so on. If both NOR gates and the associated wires have the same delays, both outputs will oscillate indefinitely with a period of 2 gate delays. In reality, the two path delays will not be identical, forcing the latch to go to a stable state. Because the final output state will vary from one latch to another, the input combination R = S = 1 is not usually applied. It is up to the circuit designer to insure that this condition never appears.

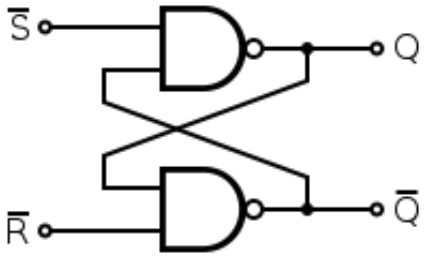


Figure 2 A NAND-based #SR Latch

An alternate model of the SR latch can be built with NAND gates, as shown in Figure 2. *Set* and *reset* now become active low signals, denoted S and R respectively. Otherwise, operation is identical to that of the SR latch. Historically, the NAND-based #SR latch has been predominant, despite the notational inconvenience of active low inputs.

Gated SR Latch

For many applications it is helpful for the latch to have a "lockdown" period during which the outputs cannot change regardless of what is happening at the inputs. During this period the output is truly "latched" to its memory value. Gated latches use a clock input (also referred to as a *gate* or *enable* input) to implement such a lockdown period. The latch inputs are ignored except when the clock signal is asserted. This effectively makes time discrete, since we do not care what happens when the clock is not asserted.

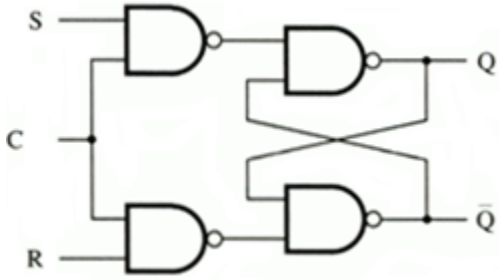


Figure 3 A NAND-based Gated SR Latch

Figure 3 shows a gated SR latch implemented using NAND gates. When C = 1, the latch is said to be *enabled*, i.e., the outputs can respond to the inputs, and the circuit behaves like an ungated #SR latch. For instance, when S = 1 and R = 0, then the upper input to the top right NAND gate is #S = 0, and that to the lower right gate is #R = 1, resulting in Q = 1 and #Q = 0.

By contrast, when C = 0, the gate is shut, i.e., the latch is *disabled*, the left pair of NAND gates keep both #SR latch inputs at 1, maintaining Q and #Q at constant values.

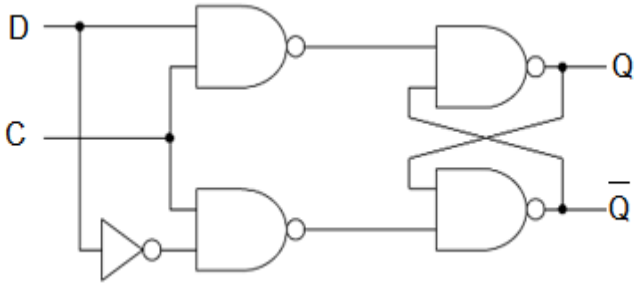


Figure 4 A NAND-based Gated D Latch

Gated D Latch

The gated D latch, shown in Figure 4, avoids the "forbidden" inputs SR = 11 using an inverter that connects the S and R inputs. Other than the gate input C, there is only one other input, D. When C = 1, this is like a gated #SR latch that has been enabled, except that the inputs to the first set of NAND gates cannot be 00 or 11. Thus a gated D-latch may be considered as a *single-input synchronous SR latch*. This configuration prevents the restricted 11 input combination from appearing. It is also known as *transparent latch* or *data latch*. The word transparent comes from the fact that, when the clock input is on, the D input value propagates directly through the circuit to output Q.

The gated D latch can be implemented in different ways. Figure 5 shows another way of building the gated D latch.

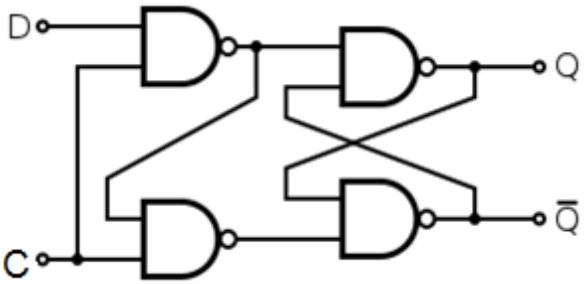


Figure 5 An Alternate NAND-based Gated D Latch

Flip-Flops

With each of the circuit designs (latches) discussed so far, the outputs can change at any time, as can the inputs. That leads to problems if the inputs switch, and then before the outputs settle into their new state, the input switches again—or if the outputs are near the "metastable" state half-way between 0 and 1 and the inputs switch to the hold values. Timing requirements such as minimum pulse width, setup and hold times, and allowance for propagation delays are impossible to guarantee with asynchronous digital logic circuits such as latches.

One way to guard against such problems is to make sure that there is ample time for all propagation delays, that input signals cannot change too fast, and that there is no chance for one of the questionable inputs to arise.

The simplest, and most common, way to do this is to make the clock input a periodic signal, and to synchronize the input signals with the clock, resulting in synchronous circuits called flip-flops.

Master-Slave D Flip-Flop

Flip-flops respond to input signals only when the clock transitions from 1 to 0 or 0 to 1. One way to implement this edge-triggering behavior is to use two D latches in series as shown in Figure 6. The clock inputs of the two latches are complementary to each other. When the clock signal is low, the second latch is opaque, and so the output Q remains constant. At the same time, the first latch is transparent and so any changes in D are transmitted to its output. At the time the clock signal transitions from low to high, the output of the first latch becomes locked and is transmitted to the output of the second latch. It is called *master–slave* because the second latch in the series only responds to changes in the output of the first (master) latch.

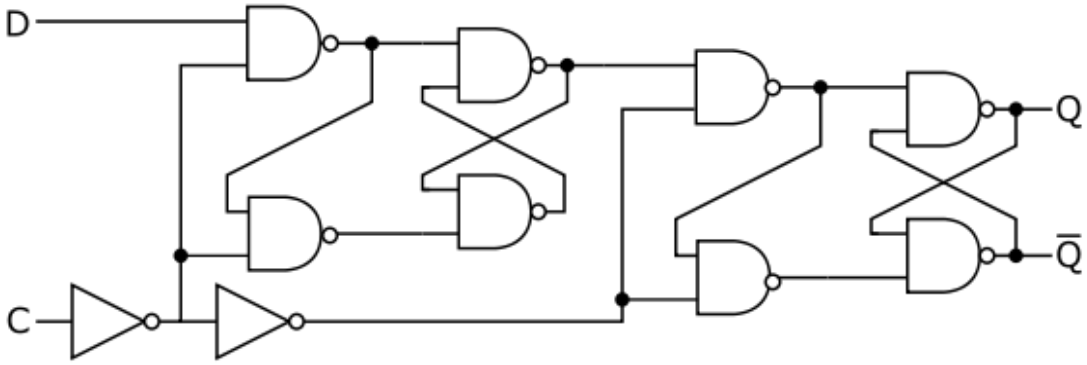


Figure 6 A Positive Edge Triggered Master-Slave D Flip-Flop

Edge-Triggered D Flip-Flop

The master-slave flip-flop is an adequate design for a D flip-flop. There are other types of flip-flops, not discussed here, for which it does not work well. The master-slave J-K flop-flop, for example, exhibits a phenomenon known as *ones-catching*. A spurious 1 on the input will be latched and propagated to the output even if the input returns to 0 before the end of the clock period.

The problems of hazards and ones-catching can be solved by designing a storage circuit that both samples its inputs and stores data based on the transition of a clock pulse. If the combinational parts of a circuit could settle during the time the clock signal was true, and the storage part of the circuit sampled the input and saved the result when the clock changed from true to false, there will be no problems with hazards at the output nor with ones catching. A storage circuit like that is called an *edge-triggered flip-flop*.

Figure 7 shows a positive edge-triggered D flip-flop that does not use the master-slave approach. It uses 3 SR latches. The bottom latch stores the D value and the top latch stores #D. The latch at the output prevents the output from changing except during a 0-to-1 transition of the clock.

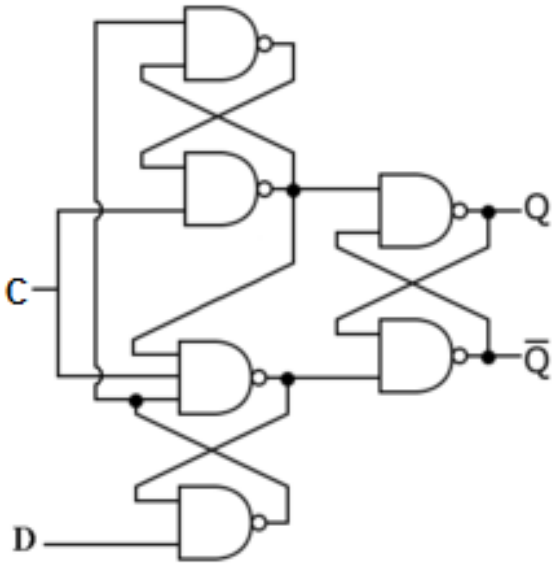
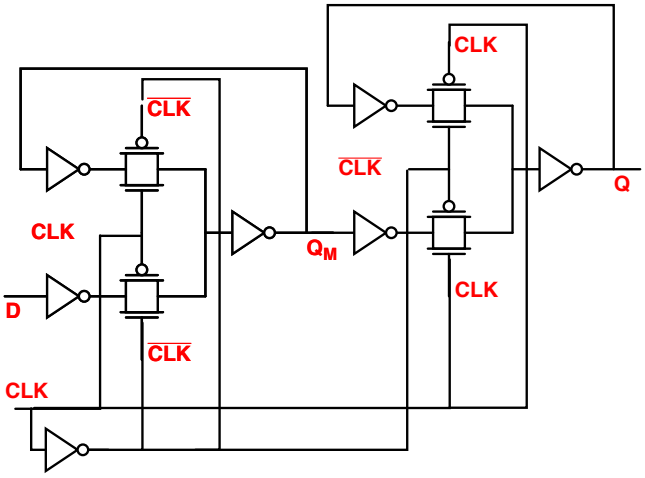
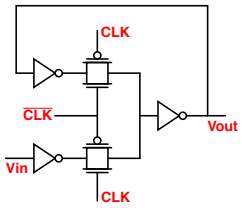


Figure 7 A Positive Edge Triggered D Flip-Flop

Additional Designs

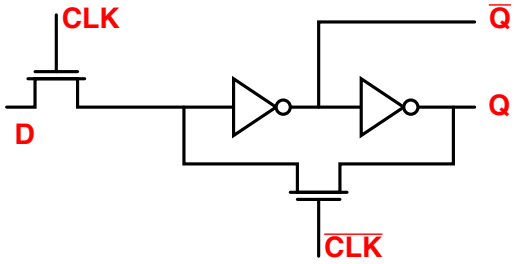
The following are additional designs, several of which are used in high-speed I/O circuits.

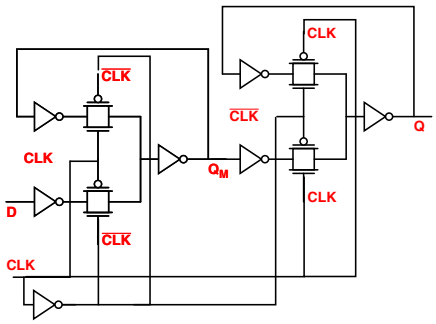




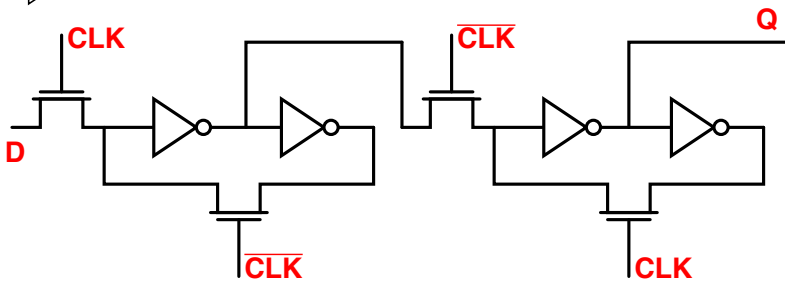
Clock network drives 4 transistors per latch … power-expensive.

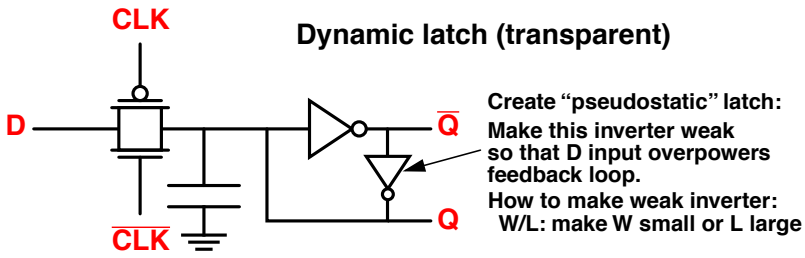
Alternative design:

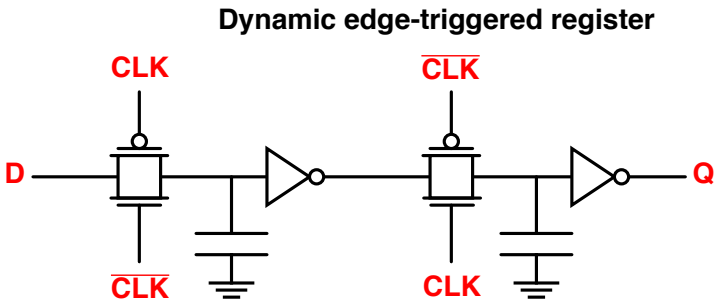




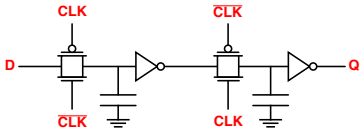
Similar example, with master-slave organzation





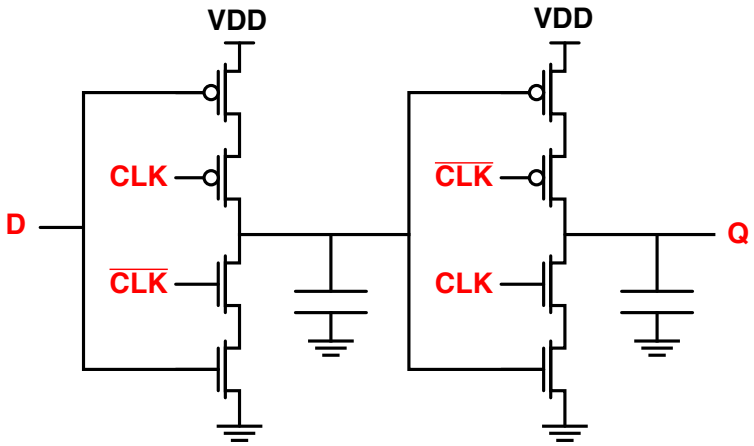


Non-overlapping clocks: Clocked CMOS

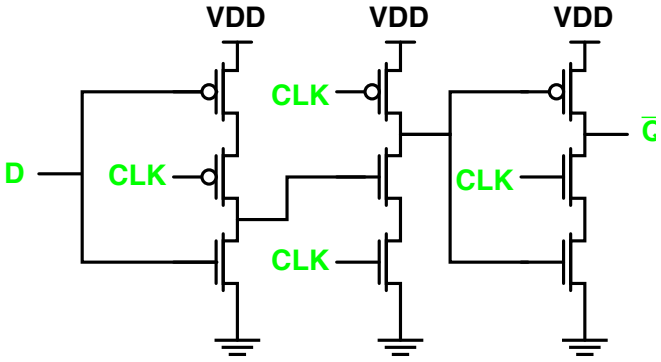


This allows feed-through when clocks overlap.

This design does not:



"True" Single-Phase Clocked Register



These examples are included just to make you aware that there are many, many types of latches out there.

Digital Logic Analyzer

The Intronix Logic Port has considerable digital signal measurement capabilities and we will learn a few of its features in this course. A picture of the logic port hardware is shown in Fig. 1. We will use the name Digital Logic Analyzer (DLA) to refer to the Logic Port.

The main difference between the acquisitions of digital signals as compared to analog signals is that for the digital work there are only two possible output states: logical zero and logical one. Thus, a digital logic analyzer only needs to use 1 bit per time step per channel whereas analog signals need multiple bits/time/channel. Each channel samples the signal on an input channel and sets the result to zero if the voltage is less than some threshold and sets it to one otherwise. This threshold can be adjusted. This approach means that you will never learn anything about rise and fall times or about any absolute voltage in a digital circuit.

One advantage of a DLA is that it can look at many signals in a circuit simultaneously. While an oscilloscope can typically look at 2-4 channels, our DLA can look at 34 channels. For the same amount of memory, DLAs can store at least 8 times as many points, so they usually have good "glitch-capture" capabilities, i.e. the ability to see transitions that happen much more rapidly than the nominal "frequency" of a clocked circuit. Mixed-Signal Oscilloscopes (MSOs) have both digital and analog channels, and are nice for digital circuits, since the analog channels can be used to obtain rise and fall time data as well as absolute voltage levels on digital signals of interest.

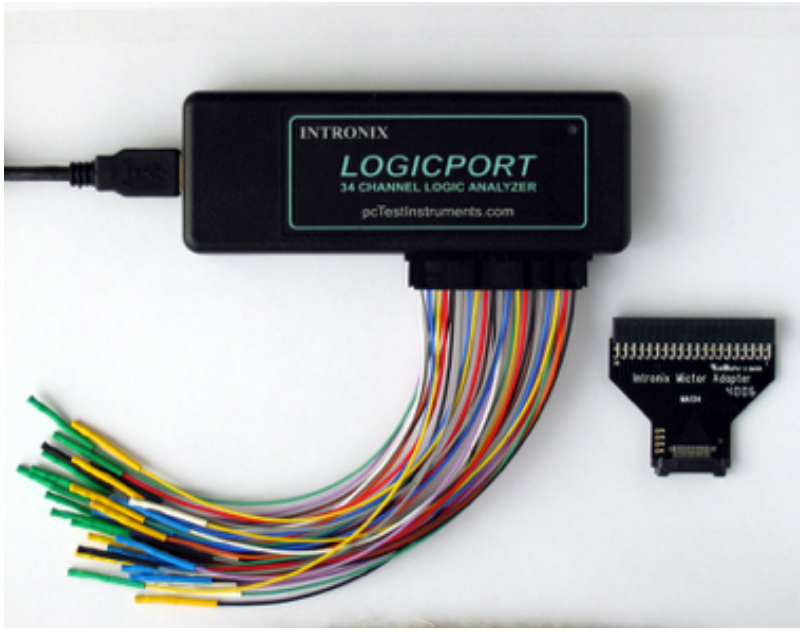


Figure 1 Intronix Logic Port

The trigger for the DLA is set in the Logic Port software. In DLA, the trigger method can be much more sophisticated than that of an oscilloscope. Like the oscilloscope, the DLA is continuously sampling data at some specified rate. When the trigger signal is identified, that time is placed at the center of the trace and the outputs before and after the trigger time are displayed on the screen. There are many ways to set the triggers. For example, you can look for a certain transition to occur or

you can look for a particular combination of states to occur in certain channels. For example, let's say you had a circuit that contained a 4-bit binary counter along with some other logic elements. Let's say you sampled 9 signals altogether, and the counter signals were on the lowest channels 0-3. If you wanted to trigger the circuit when the counter output was nine, you would specify the trigger as XXXXX1001. The "X's" represent don't cares, (and there would be up to 12 of them, depending on how many digital channels were active). This technique is known as "pattern triggering."

You can have two "levels" of triggers (this is called sequence triggering). For example, maybe you want to trigger when the output is 10XX10101, but only if sometime prior to that the output was 00000XXXX. The former sequence would be entered as the "find event" (P1) and the latter sequence would be entered as the trigger event (P2). When the first level combination is sensed, the second sequence is searched for. When the complete trigger sequence is found, the selected signals are displayed on the CRT. The middle of the display corresponds to the time that the trigger occurred, so the left side of the screen displays the signals that existed before the trigger and the right side shows the evolution of the signals after the trigger. A reset event can also be specified, and if this event occurs after the pre-trigger event was detected but before the trigger event happened, the trigger is reset to its initial state and the DLA waits for the pre-trigger signal again.

Standard edge-triggering can also be used, and for pattern triggering a single transition (up or down) for one of the signals can be specified. Edge triggering can be specified as the type of events searched for during sequence triggering.

Pre-Lab Preparation

Part I – SR Latch

Design a gated SR latch using either NOR or NAND gates.

Draw the logic diagram and wiring diagram for the latch using the available CMOS chips.

Use PSpice to simulate the SR latch and plot the output and the input signals as a function of time when you use a synchronous mod-4 counter to generate the input signals.

Part II – Master-Slave SR Flip-Flop

Design a master-slave SR flip-flop using the SR latch designed above in Part I.

Draw both the logic diagram and wiring diagram for the slip-flop using the available CMOS chips.

Use PSpice to simulate the flip-flop. Use a synchronous mod-8 counter to generate all possible combinations for the Set and Reset signals, including the SR = 11 combination. Use the two most significant bits of the counter output for the Set and Reset signals. Use the same clock to drive the counter and your flip-flop, but invert the clock before connecting it to the flip-flop.

Part III – Edge-Triggered Flip-Flop

Design an edge-triggered register (e.g., positive or negative edge-triggered D-type flip-flop, positive or negative edge-triggered T-type flip-flop, or positive or negative edge-triggered JK-type flip-flop).

Draw both the logic and wiring diagrams.

Use PSpice to simulate the flip-flop. Use a 100 kHz "digclock" with a duty factor of 50% for the clock signal and a 50 kHz "digclock" with a 70% duty factor as the input to the flip-flop (for the JK flip-flop, tie the J and K together). Plot both clocks and the output of the flip-flop versus time.

Pre-lab Questions

Attach the results of your pre-lab work (drawings, simulations, etc.) to this question sheet.

1. Explain the difference between a latch and a flip-flop.
2. Why is the condition S = R = 1 not allowed for an SR latch?
3. There is no such thing as an ungated D latch. Why?
4. What are the differences between the D, T, and JK flip-flops?
5. Did you observe any glitches in the simulation results? If so, where?
6. What are *setup* and *hold* times for a flip-flop? Why are they important?
7. What is a master-slave flip-flop?

In-Lab Procedure

Bring flash drives to store your traces.

Ask the TA questions regarding any procedures about which you are uncertain.

Turn off all power supplies any time that you make any change to the circuit.

Do NOT apply more than 5 V to the circuit at any time.

Arrange your circuit components neatly and in a logical order.

Compare your breadboard carefully with your circuit diagram before applying power to the circuit.

Complete the following tasks:

Part I – SR Latch

1. Construct the SR latch.
2. Manually test all possible (4) input combinations with clock C = 1 and record both the input and output voltages with a DMM.
3. Set the clock high, tie S and R together to the output of the function generator "sync out" and set the frequency to 100 kHz. Observe the output on a DLA and note any irregularities.

Part II – Master-Slave SR Flip-Flop

4. Construct the master-slave SR flip-flop.
5. Set the function generator to 100 kHz and use it to drive both your SR flip-flop and a synchronous mod-8 counter. Use the two most significant bits for the S and R inputs of your flip-flop. Plot the clock, Set, Reset, and the two SR flip-flop outputs on the DLA.
6. Measure the time delay between the clock and the output signal, as well as the delays between the clock and the intermediate switching circuit signals.

Part III – Edge-Triggered Flip-Flop

7. Construct the edge-triggered flip-flop.
8. If using a JK or T flip-flop, tie the inputs to 5 V. Otherwise, tie the input to #Q. Drive the flip-flop clock with a 100 kHz square wave.

9. Plot the clock, the input, Q, and #Q simultaneously on the oscilloscope.
10. Measure the delay time of the output on the oscilloscope. Record the output voltage for both the 0 and 1 states.
11. Increase the clock frequency and note if and when the circuit fails.

Helpful Hints

You do not need to design the circuits of this lab from scratch; you should be able to find the designs in a digital logic textbook.

You can make inverters by tying together all of the inputs of a NOR or NAND gate. This may be a useful way to reduce the chip count.

Try to arrange your components on the breadboard in a similar arrangement as on your wiring diagram.

Post-Lab Report

Write up your circuit, schematic, and lab procedure. Mention any difficulties encountered during the lab. Describe any results that were unexpected and try to account for the origin of these results (i.e. explain what happened).