

Lab #4 Sequential Logic Design

Objective:

To study the behavior and applications of flip flops and basic sequential circuits including shift registers and counters.

Preparation:

Read the following experiment. It will save time if you enter the circuit designs for Parts 3, 4 and 5 using the graphic design editor before coming to the lab. Bring your textbook to the lab!

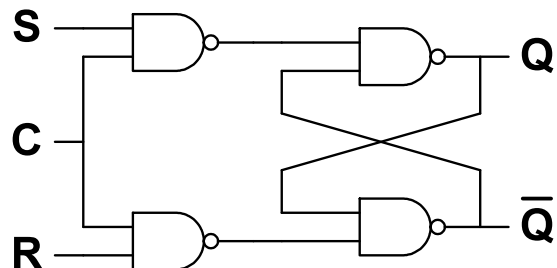
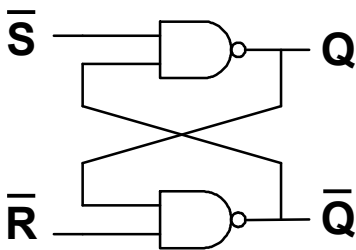
Devices used:

sn74LS00 Quad 2-input NAND
sn74LS76 Dual JK Master-slave flip flop
2732A EPROM
MAX3000 EPM3032ALC44-10 CPLD
Digital-to-Analog converter (DAC)
Op Amp

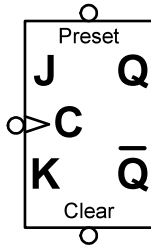
Experiment:

Latches and Flip-flops

1. The RS latch is the fundamental element in sequential logic design. Construct the circuits for the RS latches shown below using the 7400 TTL package. Using switches and LEDs, carefully verify their operation and write their function tables.



2. The JK flip flop is a widely used Master-slave device that features in the design of many sequential circuits. It is versatile in that it may be easily used as a toggle flipflop (T-type) or a D flipflop (D-type)



- (a) Carefully verify the behavior of the JK flipflop (in the 7476 TTL package) and write out its characteristic table. Note carefully the clock edge to which it responds as well as the asynchronous operation of the preset and clear inputs. Use toggle switches for the inputs and LEDs on the outputs.
- (b) Connect an inverter between the J and K inputs and verify that the operation is that of a D flip flop. Write its characteristic table.
- (c) Connect the J and K inputs together and verify that it operates as a T flip flop and write its characteristic table.

Registers and Counters

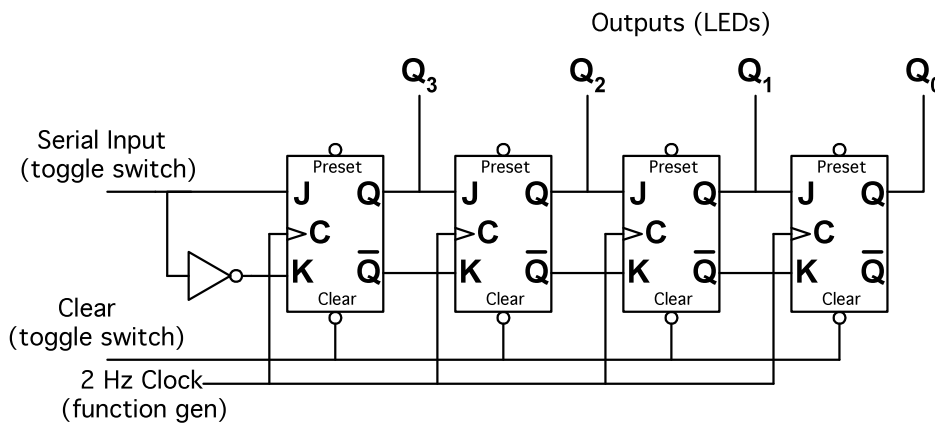
A register is a group of n flip-flops operating together whose outputs are considered to represent an n-bit binary number. A single flip-flop can be considered to be a single-bit register. The simplest form of a register is used to just capture and hold a value applied to its inputs (eg. the D inputs of D flip-flops) in response to a "write" command (the clock pulse). However, registers may also be considered to be *operational* meaning that the value they hold may be altered in response to an external input. For example, an n-bit counter may be considered a register with the capability to increment (or decrement) its value. Another common example is shifting in which the values of individual bits are passed from one position of the register to another. Shifting all bits one position to the left multiplies the register contents by 2; each shift to the right divides the register value by two. We will study the behavior of

some examples below. Registers can be forced to zero by the simultaneous application of a pulse to the asynchronous clear inputs of each flip flop that comprises the register.

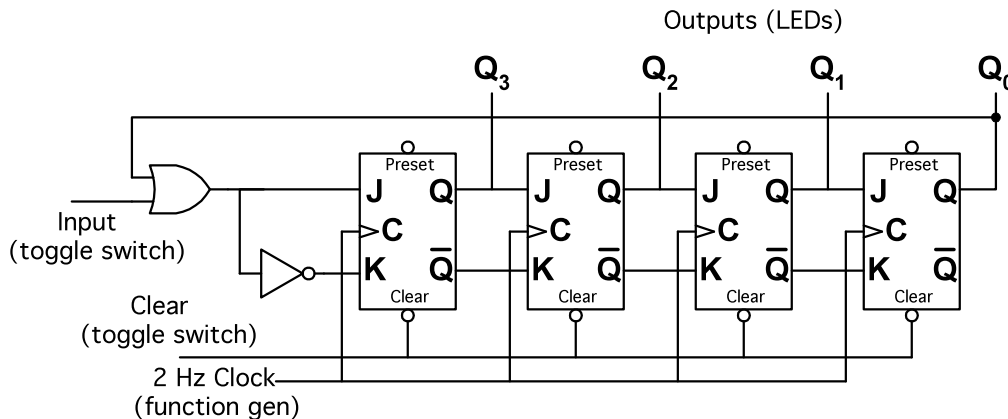
For the remainder of this lab, design the required circuit using the graphic design editor (Appendix B, page 832) included in the Quartus II design package. For each circuit, load into the MAX3000 chip and test on the bread board using LED's and switches.

3. *Shift registers*

(a) Flip-flops can be connected in series to form shift registers in many ways. A simple left-to-right shift register can be constructed from JK flip-flops configured as D flip-flops as shown in the circuit below. Note that an inverter is required only for the left-most flip-flop; thereafter the Q' output is available instead. The preset inputs should be tied to logic HI. Experiment with this circuit using a low frequency clock input provided by the function generator so that you can observe the register output patterns $Q_3Q_2Q_1Q_0$ on LEDs. Using toggle switches to provide the clear input as well as the serial input. Note that the JK flip-flop in the editor does not contain the output Q', but it can be obtained by inverting Q (attach an inverter to Q and use the output of the inverter as Q').

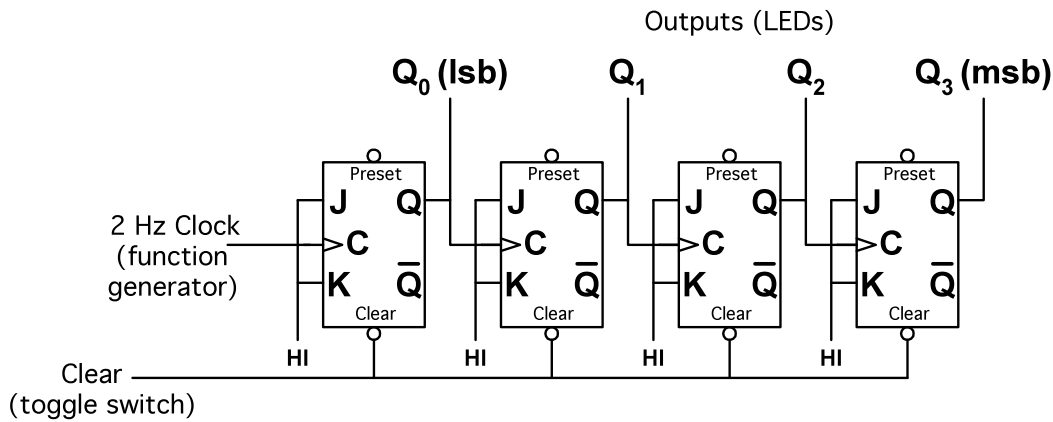


(b) Modify your circuit from part (a) as shown below to implement a "rotate" capability in the register by connecting the Q_0 output back to the input through an OR gate. This is also known as a *ring counter*. The serial input of the register is done with the toggle switch via the OR gate.



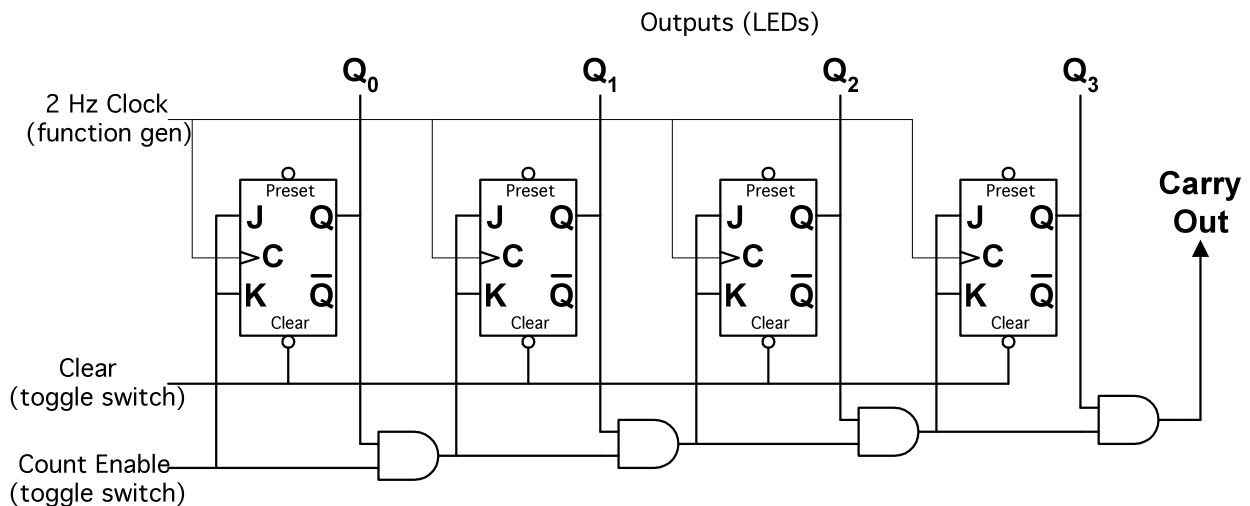
4. Asynchronous counter

Implement the following circuit that uses JK flip-flops configured as T flip-flops ($J = K = 1$). Note that the clock input to the flip-flops for Q_1 , Q_2 and Q_3 come from the output of the preceding stage; ie it is asynchronous because the changes in the output bits representing the state are cascaded in time. For this reason it is also known as a *ripple counter*. Write the sequence of output states observed on Q_3 , Q_2 , Q_1 and Q_0 starting with all flipflops cleared.



5. Synchronous counters

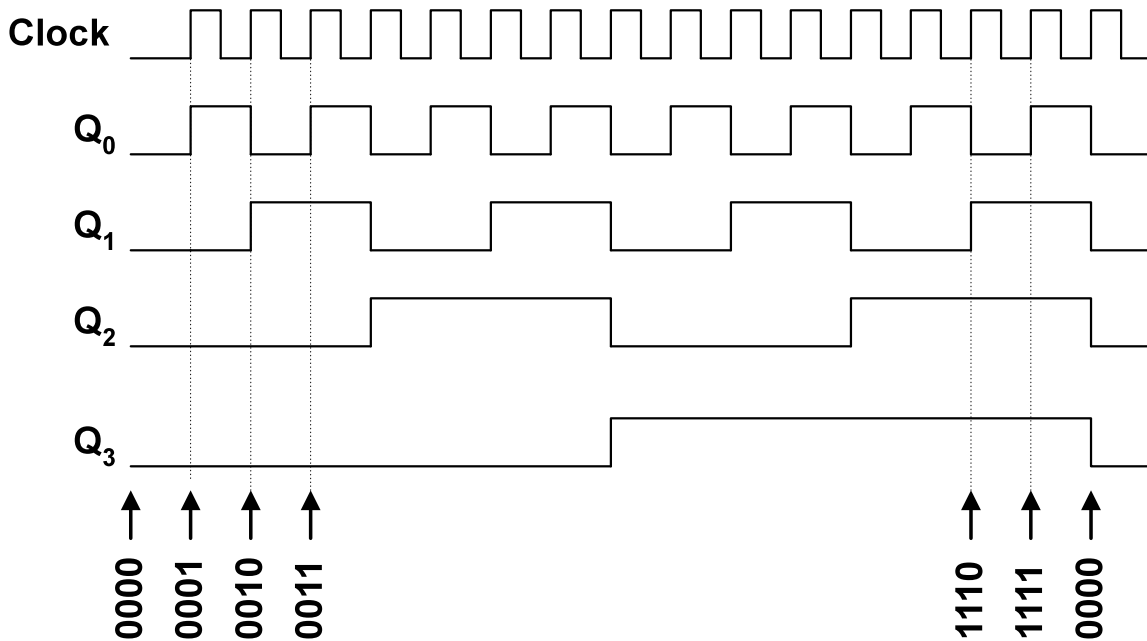
An asynchronous counter can cause difficulties due to intermediate (short) output states caused by the time delays. For example, in changing from **0011** to **0100**, the count value actually undergoes the sequence **0011** \rightarrow **0010** \rightarrow **0000** \rightarrow **0100** where the states **0010** and **0000** exist only for the time for the gate delays through a single flip-flop. The bits change in the order of LSB to MSB. Similarly, changing from state **0111** to **1000** gives short intermediate states at **0110**, **0100** and **0000**. A better design is the *synchronous* counter in which all flip-flops receive a common clock pulse:



Build the circuit for the synchronous up-counter as shown. Verify its timing and operation using the timing diagram shown below. Note that the count enable can be used to freeze the count at any time and that count transitions occur on positive edges of the clock. Increase the clock frequency to 100KHz and measure the frequency at each output:

Frequency measured at: Q₀: _____ Q₂: _____
 Q₁: _____ Q₃: _____ Carry Out: _____

The design of this counter uses *serial gating* which means that, for example, in going from state **0111** to **1000**, there are three delays through the AND gates required to generate the T-input to the final bit. This restricts the maximum count frequency. *Parallel gating* is analogous to look-ahead carry logic and allows for the design of faster counters.



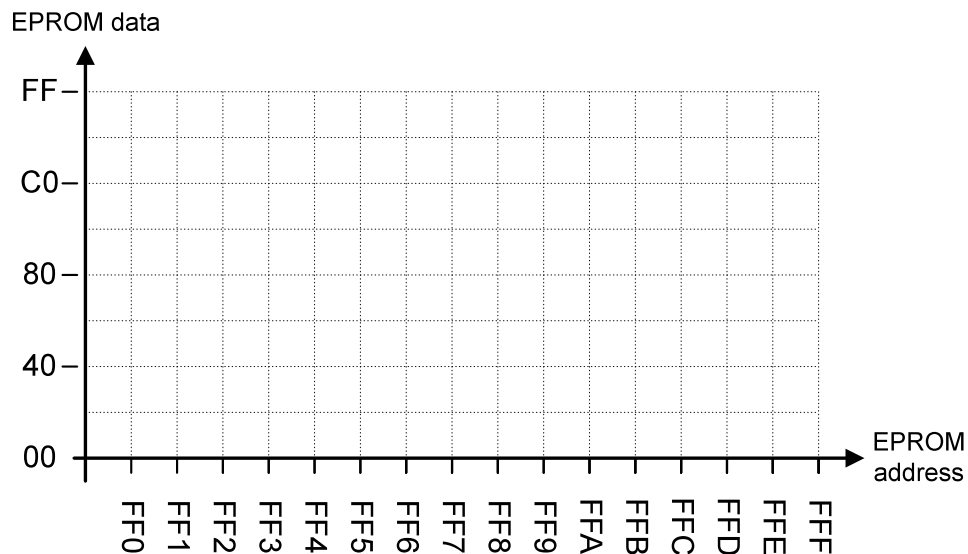
6. Simple Function Generator

In this part of the lab we will use the 4-bit synchronous counter to address an EPROM to implement a simple function generator.

First using the toggle switches, examine the data that is stored in the top 16 locations of the EPROM and complete the following table. Note that these are the addresses 1111 1111 0000 ($FF0_{16}$) to 1111 1111 1111 (FFF_{16}).

Address (A0-A11)	Data (O0 - O7)	Address (A0-A11)	Data (O0 - O7)
FF0		FF8	
FF1		FF9	
FF2		FFA	
FF3		FFB	
FF4		FFC	
FF5		FFD	
FF6		FFE	
FF7		FFF	

Now, plot the data from the table, on the following grid:



Notice that the 8-bit data is a set of 16 samples that form one cycle of an approximated sine wave pattern. In fact, by simply reprogramming the EPROM we could generate any arbitrary waveform. By repetitively accessing each of the 16 sample values, the outputs of the EPROM would thus represent a periodic function, in this case the sine function. Increasing the speed at which we cycle through the

samples would increase the frequency of our sine wave output. This is, of course, the function of the counter.

We can easily generate the 16 addresses required to repetitively access all 16 samples by driving the bottom four address inputs with a binary counter while the top 8 address lines are held HI. This will continuously generate addresses from $FF0_{16}$ to FFF_{16} as shown in the circuit below. To observe our fabricated waveform on a 'scope, we convert the 8-bit digital samples on the output of the EPROM to analog by way of a Digital-to-Analog Converter (DAC). A DAC produces an analog voltage output that is proportional to the digital input. In this case, the minimum voltage output (0V) is given by digital input 00000000 and the maximum voltage output (+5V) is given by digital input 11111111. You will learn much more about the internal design and operation of DACs in later courses.

Connect the output of the counter from part 5 of this lab to the lowest four address inputs of the EPROM as shown below, leaving the carry output unconnected. Connect the 8 data out lines of the EPROM to the DAC. Vary the counter clock frequency and observe the effect on the waveform that is generated.

What is the peak-to-peak magnitude of the sine wave that is generated ? _____

What sine wave frequency is generated with the 50KHz counter clock ? _____

How is the sine wave frequency related to the counter clock frequency ? _____

Suggest ways that the waveform could be made "smoother".

