Chapter 5: More Counters and Registers

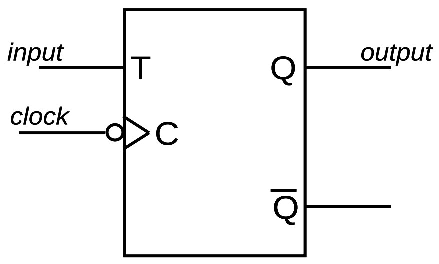
This week we take a look at the fundamental circuits built with flip-flops which are used for memory, counters, and registers.

I. Counters

Last week we introduced some basic counting circuitry which could be used for dividing the frequency of clock signal. These basic flip-flop circuits can be generalized to count and output the number of edge triggers received.

The basic counting component is the T-type flip-flop, which is just a JK-type flip-flop with the J and K inputs tied together to form a single input. The symbol and truth table for the T-type flip-flop are shown in figure 1 to the right. The operation of a T-type flip-flop is very simple: On receiving a clock-edge, if the input is LOW then the output stays the same, but if the input is HIGH then the output flips its value.

As we have seen with Verilog and FPGAs already, counters are critical components in most useful digital circuits. Just like all of the other useful circuits we have seen so far, counter circuits come as a complete IC package. There are two basic counting architectures: ripple counters and synchronous counters.



Logic table

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Figure 1: *negative-edge triggered T-type flip-flop.*

Ripple counters

The ripple counter is the simplest of the two types of counters and is shown in figure 2 below. The counter consists of a series of T-type flip-flops with the output of the previous flip-flop connected to the clock input of the next flip-flop. The T-inputs of each flip-flop are held high, so that on each clock trigger, the flip-flops change state.

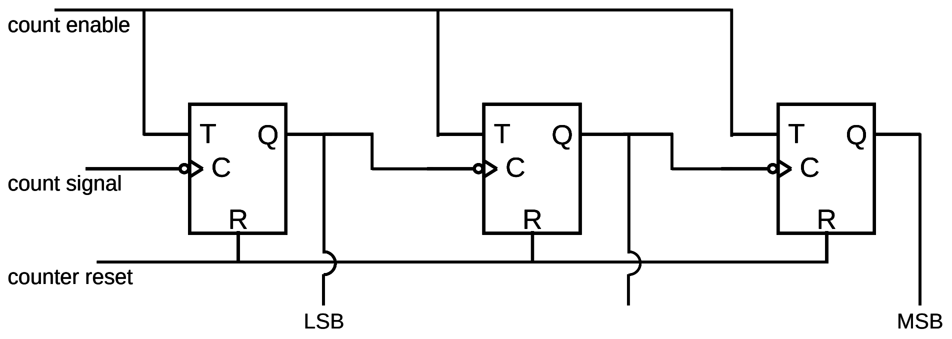


Figure 2: *3-bit ripple counter with reset-to-zero and count enable features.*

When a flip-flop with state LOW receives a negative-edge clock signal, it flips its state and output to HIGH, however the next flip-flop does not register the count because it must wait for a falling-edge, which will only occur when the initial flip-flop receives a clock trigger and returns to the LOW state. In this manner the counts propagates from one flip-flop to the next. The circuit is called a ripple counter, because the count signal must propagate sequentially from one flip-flop to the next for it to be correctly stored. For example, if the counter went from 011 (i.e. 3) to 100 (i.e. 4), then the first flip-flop changes from 1 to 0, followed by the second flip-flop changing from 1 to 0, and finally the third flip-flop changes from 0 to 1.

Ripple counters suffer from significant propagation delays. This means that you have to wait for the counter to update before sending another clock pulse. Even worse, the bigger the number you want to count to, the longer you have to wait for the counting pulse to propagate through all the flip-flops. Ripple counters are easy to explain, but are not used very much because of their significant ripple delay.

Synchronous Counters

Synchronous counters, as their name implies, update all their flip-flops simultaneously. Figure 3 below shows the circuit diagram for a synchronous counter. The counting clock signal is sent to the clock input of all the flip-flops, and AND gates are used to determine if a flip-flop should change its state or not.

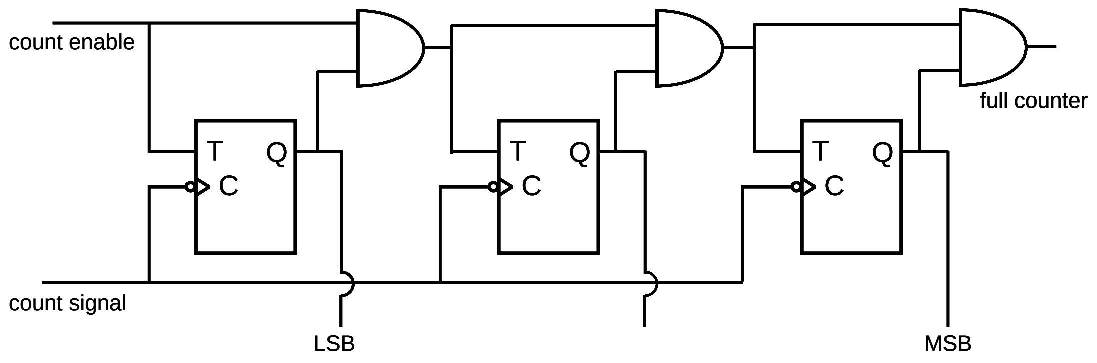


Figure 3: *3-bit synchronous counter circuit with T-type flip-flops. The counter reset-to-zero feature shown in figure 2 has been omitted for clarity.*

Synchronous circuits are little harder to understand because one has to keep track of the state of the AND gate, but they are standard type of counter found on IC chips. The synchronous counters of the 74LS family which are available in the electronics lab include the '90, '92, '93, '160, '161, '191, '193, '269, and '393.

II. Registers

A register is a general type of memory. While counters store a number and increment it by one on each clock cycle, a register can store any series of bits and present it at its output when requested. Registers are the basic type of memory used in digital circuits and computers.

The simplest register is the Parallel In Parallel Out (PIPO) register, which is just a series of independent D-type flip-flops with a shared clock input. Figure 4 below shows the circuit diagram for a PIPO-type register.

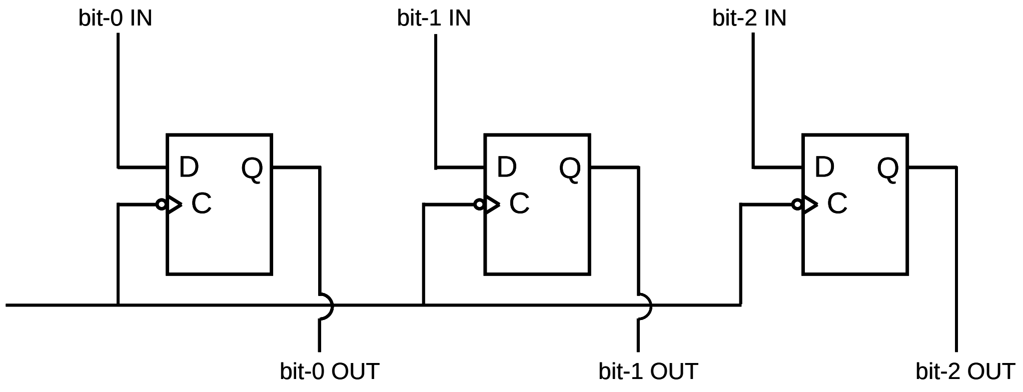


Figure 4: *Circuit diagram for a 3-bit PIPO-type register.*

PIPO-type registers are very fast since each stored bit of information can be retrieved independently of the other. In practice, most memory is not of the PIPO-type since it requires a very large number of input and output lines. For example, if you wanted to store 1 Megabyte of information or 8 Megabits (something which your computer does on regular basis) then you would need a total of 16 million wires.

Shift-Registers

Shift-registers are a common type of memory register. In the example circuit of figure 5 below, the bits of information are presented to the register one-after-the-other on the falling-edge of the clock at the serial input. If necessary, the bits of information can be retrieved in parallel, but normally they are retrieved one-after-the-other in the same order that they were stored at the serial output. The circuit is an example of a First In First Out shift register or FIFO-type register.

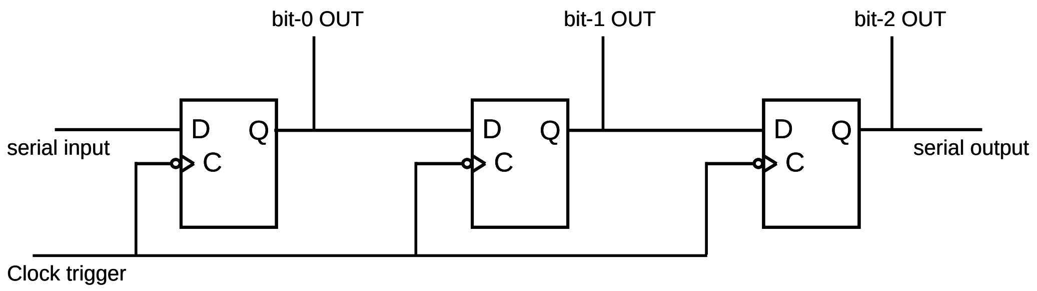


Figure 5: *3-bit FIFO-type shift register with a parallel out read option.*

The main advantage of shift-registers is that they require only a single input and output line, so that to store 1 Megabyte of information only require one input and one output wire and of course 8 million D-type flip-flops. The disadvantage of FIFO-type shift registers is that they are slow, since they require as many clock cycles as bits of information that they store, and the memory is erased when it is read on the serial out line. FIFO shift-registers are also used for converting data from serial format to parallel format.

Design Exercises

Design Exercise 5-1: Create a Quartus II project based on design exercise 4-5 (last week) that will take a 15-bit binary number from the switches on the DE2 (SW[0-15]) and displays the number in decimal representation on the LED "Hex" displays (HEX0[0-6], HEX1[0-6], HEX2[0-6], HEX3[0-6], HEC4[0-6]).

Design Exercise 5-2: Create a Quartus II project that will count the number of positive edge triggers on an input pin for 100 ms (and every 100 ms after that) and display the result in little-endian binary format with red and green LEDs (LEDR[0-17] and LEDG[0-8]). How many bits must your counter use to measure all the counts from the 27 MHz clock on the DE2?
Hint: Use two always blocks for your Verilog program/circuit.