FEATURE

Computer Logic Design Part 3

This month we'll begin to look at flip-flops, the beginning of memory logic. A fundamental area of computer design, memory and counting move into areas of logic without finite states.

Steve Rimmer

In the first two installments of this series we looked at the rudiments of finite state logic. In this sort of logic design, any arrangement of logical elements, however huge, complicated or badly soldderered, will have a finite number of output states for a finite number of input states. Such a a logic array must be essentially static... it doesn't do anything until you change something at one of its inputs.

This month we're going to introduce a new element into logic design, this being the D flip flop. Flip flops have two states, just as do other logic elements, but their output states are dependant upon what happened to their inputs over time.

Remember This

A D flip flop has six connections. These are its input, its clear and preset lines, its clock line and its two outputs, Q and $\underline{Q}$. The state of Q will always be the compliment of the state of $\underline{Q}$.

The simplest function of a flip flop is to remember the state of its data input after the state has changed. If you set the data input either high or low and send a positive pulse to the clock input, the state of the input will appear at the Q output. It will remain there after the clock pulse has gone, and it will remain even if the data input changes.

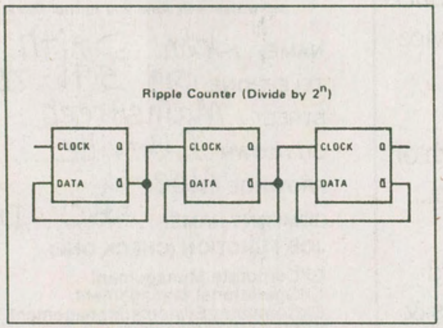

Fig. 1. Type D flipflops connected as a ripple counter.

All by itself, one flip flop is not terribly useful, but they have all sorts of applications when they get together and gang up on logic design problems.

Figure One illustrates a D flip flop "ripple" counter, or binary divider. This will divide the frequency of the pulses at the leftmost clock input by 2 raised to the power of the number of flip flops in the counter.

Figure Two illustrates a shift register. This is probably a more useful example of what flip flops do. Let's assume that there's a high state at the input to the shift register to begin with. The first time a clock pulse is applied to the common clock line, the high state will appear at the Q output of the first flip flop. The next time a clock pulse occurs, it will appear at the Q output of the next flip flop, and so on. With each successive clock pulse the initial state... the data... will shift itself one stage further along in the circuit.

The ripple counter in figure one can be thought of as a frequency divider, but it's really a binary counter. If you regard the Q outputs of the three flip flops as being bits, it will cycle through the binary numbers from zero through seven as pulses are applied to its input. The leftmost flip flop represents the lowest order bit. If you were to add more flip flops to the circuit, it would count up to larger numbers before wrapping back to zero.

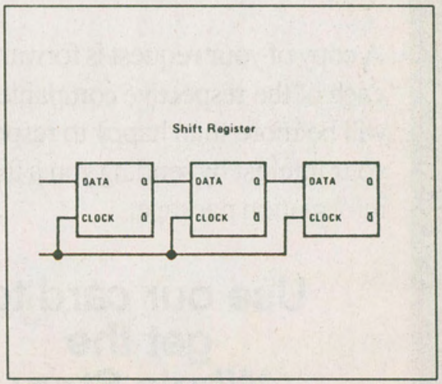

Fig. 2. Type D flipflopds connected as a shift register.

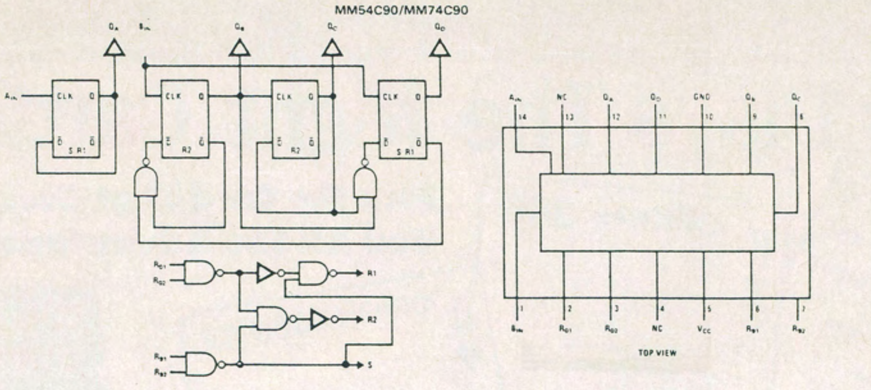

Fig. 3. The 74C90 4-bit decade counter. All illustrations are courtesy of National Semiconductor.

Count to Ten

Computers think in numbers which are even powers of two. People like to think in numbers which are even powers of ten. Ten is really a very troublesome number to use as a base, and it's only our finger count which has prompted us to settle on it.

In designing computer circuitry, we invariably use the numbers which best suit computers. However, specialized logic circuits which are to be used by people often require some consideration of their numerical preferences. In these cases, it's useful to know how to make a circuit which wants to count up to sixteen behave itself and count to ten.

A four stage binary counter will have sixteen output states, that is, sixteen possible values represented by the four Q outputs of its flip flops. Of these, the ones representing zero through nine are of interest in a circuit which will count to ten. If you wanted to build a frequency counter, for example, you'd only want to count to ten for each decade.

The problem, then, is to make the sixteen state counter stop at ten and wrap itself back to zero. The problem is very simple, but it requires that we use an as yet undiscussed line of the flip flops. If you pulse the clear line of a flip flop low, the flip flop will forget whatever its has been told previously. The Q output will go low and the Q output will go high. The flip flop will go to its natural quiescent state. The preset line will do just the opposite.

In a four bit counter, clearing all four flip flops will set the output of the counter to zero. It would seem that in order to make the counter count to nine and then reset itself to zero, all we need do is to hit the clear line on the tenth input pulse.

In order to detect the tenth pulse, we must watch the output of the counter with an array gates to detect the binary number ten. This number is

1010

in binary, that is, eight plus two.

Figure Three illustrates a commercial decade counter. You can see how its four flips are gated in order to make them reset when they reach a value of ten.

The clear pulse for this counter also represents the input pulse for the next counter if you want to count to numbers larger than nine. If you were to place a second counter beside the first one, with its input being the clear pulses from the first counter... inverted, in this case... its value would be the tens in the number being counted. You can add as many decades as you need to count to the number you have in mind.

We'll look at some practical applications of flip flops in the next installment of this series. ■