DESIGNER'S NOTEBOOK

Extending the counting range of the 4017

ROBERT GROSSBLATT

THE 4017 IS ONE OF THE MOST POPULAR OF the CMOS MSI (*M*edium *S*cale *I*ntegration) counters. Because it sequences its outputs one at a time, it's ideally suited to serve as a frequency divider, pulse delayer, and so on. That IC even has a carry output that allows you to cascade as many 4017's as desired.

There is an application, however, where the use of the 4017 isn't quite as simple and straightforward. We're talking about connecting several of them together so that they will all sequence their outputs one IC after another. The carry output isn't any good here because it goes high for one-half of the IC's full count and low for the other half. That's great if you want a squarewave whose repetition rate is one-tenth of the input clock frequency. But it doesn't help at all if you need a simple circuit that will take an input-clock signal and turn on more than ten outputs in sequence. Doing that requires a bit of external gating.

This month we will look at how you can arrange three 4017's to sequence in turn and provide up to twenty-five outputs with a minimum of external gating. We will be using a 4011 quad-NAND gate to achieve the desired gating arrangement. The same principle can be followed if you need more than twenty-five outputs; just add more 4017's and a few more gates. Since there are four NAND gates in the 4011, it will accommodate two more 4017's to give you a total of 41 outputs. We're losing five possible outputs for a variety of reasons that will become clear shortly.

How it works

In Fig. 1, the clock inputs of all the 4017's are tied together so that they can all be triggered by a common clock pulse, and the same goes for the reset pins. The key to making the circuit work is using the gates to control the enable pins (pin 13) so that the three IC's can be made to count in sequence. Since the enable pins are active high, they have to be brought low. We'll use the NAND gates to control the order in which the 4017's are enabled.

When power is first applied to the circuit, capacitor C1 generates a reset pulse that forces all the 4017's to output a "0" at pin 3. One of the unfortunate features of the 4017 is the lack of a convenient inhibit pin that we could use to turn off all the outputs. When the enable pin is brought high it only disables the clock input—it doesn't do anything to the outputs. That's why we need the external gating and that's also the reason that we can only get 25 outputs from the circuit instead of 30.

When the counting first starts, pin 11 of all the counters is brought low. Because pin 11 of IC1 is connected to its enable pin, the IC is enabled and starts to count. That same low signal is fed to one leg of NAND gate IC4-a at pin 5 to control the enable pin of IC2. The other leg of the NAND gate, pin 6, is connected to IC2 pin 11 through IC4-b, which is used as an inverter. Because IC2 was reset at the start of the counting process, its output at pin 11 is also low. That low output is inverted by IC4-b and the resulting high output is applied to IC3 to disable it. The high output from IC4-b is also applied to IC4-a, which in turn outputs a high to the enable pin of IC2 disabling it too. That means that only IC1 is enabled at the start of the count.

When pin 11 of IC1 goes high, it disables IC1 and at the same time that high is fed to pin 5 of IC4-a. Because both inputs to IC4-a are now high, it's output goes low and that in turn enables IC2. When the output at pin 11 of IC2 goes high, it's inverted by IC4-b and the resulting low enables IC3. The same low is applied to IC4-a causing it to change states (go high), thus disabling IC2. After nine more clock pulses the output of IC3 at pin 11 goes high applying that signal to all of the reset pins simultaneously, causing all the 4017's to reset to zero and start the whole sequence all over again.

Essentially, what we're doing is using pin 11 of each 4017 to control the enable pin of the next IC in line. When the output at pin 11 of the last IC in the sequence goes high, a high is applied to the reset pins of all the IC's causing them to be reset to the beginning. The same principle can be used to extend the sequence. Just remember that you can only use eight of the 4017's ten outputs—pin 3 is needed because there's no inhibit control and pin 11 controls the next 4017.

Let us know if you can figure out a way to use a simple gating arrangement to recover the use of some of the five lost outputs in the circuit. After all, there's always a better way to do something. **R-E**

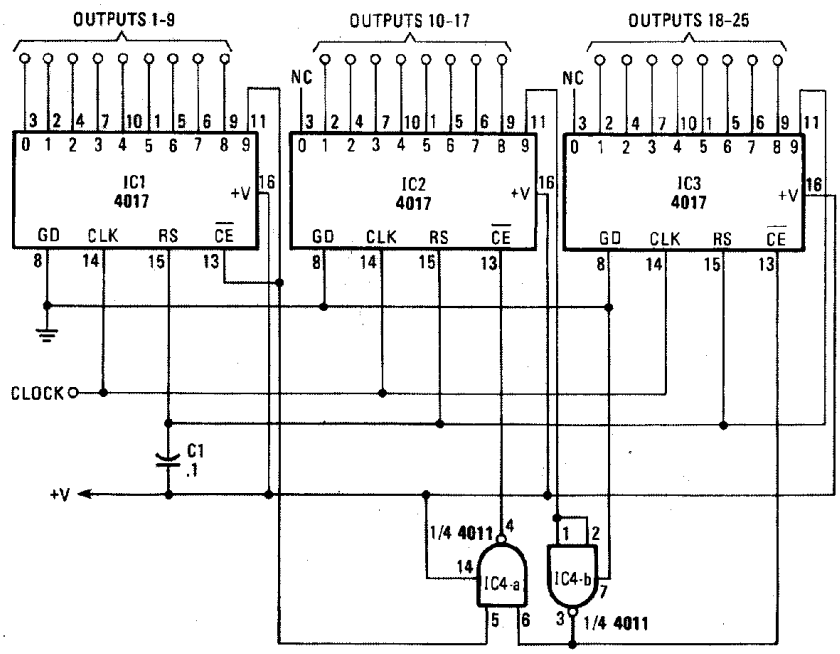

FIG. 1

THE DRAWING BOARD

Solving the reset problem

ROBERT GROSSBLATT

THE MOST IMPORTANT THING WE LEARNED from last month's discussion of counters in general and the 4017 in particular is this: you get what you pay for. Using the 4017 to get nice cheap frequency division was like a lot of things in life—it seemed like a good idea at the time but when we put the IC to work we only got half of what we expected. The circuit was certainly cheap enough but the results were a far cry from nice. Two major problems showed up that really limited the usefulness of the circuit. The first, asynchronous reset, introduced some unpredictability in the output. The second problem was that the output duty cycle would change as we changed the number we were dividing by. Now, for some applications those may not be important but for others, they can be a real problem.

Not only that, but it's a good rule of thumb in design to limit the times you're willing to shrug your shoulders and compromise. After all, one of the main reasons you're designing something from scratch is to have the circuit do exactly what you want it to do. There are already more than enough times in life when you have to meet something halfway.

The reset problem

Let's tackle the reset problem first. In a nutshell, asynchronous reset means that the IC will reset itself whenever the reset pin is brought high. Not only is the operation independent of the input clock but you also have no control of the time the RESET pin remains high. Fortunately, that problem can be licked with a little bit of imaginative gating.

The trick to adding synchronous reset to the 4017 is being able to control the RESET pin. We need some sort of gating arrangement that will make it go high when we want; and more important, make it go low when we want. We also need some way of making sure that latter signal comes when RESET is completed. Remember that the 4017 is disabled as long as the RESET pin is held high.

Let's digress for a bit. What we're talking about here is a gating arrangement that has two independent inputs and whose output will change state when triggers are applied to each of the inputs. So, as you've probably guessed by now, we need a flip-flop. Now, you can use some standard-type flip-flop such as the 4043 or 4044; but in line with the established traditions of this column, let's see what we can do with a bunch of simple gates.

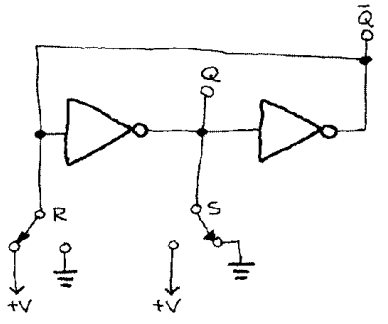

FIG. 1

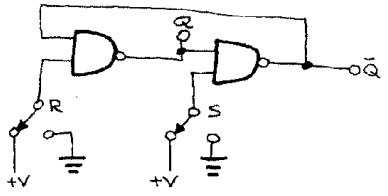

FIG. 2

FLIP-FLOP TRUTH TABLE

R	S	Q	Q̄
LOW	LOW	No Change	
LOW	HIGH	HIGH	LOW
HIGH	LOW	LOW	HIGH
HIGH	HIGH	Not Allowed*	

*See text

Figure 1 shows how we could build a simple flip-flop. It's made from two inverters and is mechanically triggered. If we connect the R switch to ground the Q output will be positive and the Q̄ output will be negative. In fact, triggering either of the outputs will produce entirely predictable results and obviously stable outputs. The only thing we can't do is connect both the R and S inputs to the same polarity—if we did, the whole thing would obviously go up in smoke. Since we all know that if something bad can happen it *will* happen, we'd better look further for our needed flip-flop.

In Fig. 2, we've done the same sort of thing with a pair of NAND gates. The operation of the circuit is more interesting and, ultimately, more useful. Let's assume the R input is connected to +V and see what happens as we switch the S input between +V and ground.

If we connect the S input to ground, the Q̄ output will be high because a NAND gate has a high output if one or more of its

continued on page 93

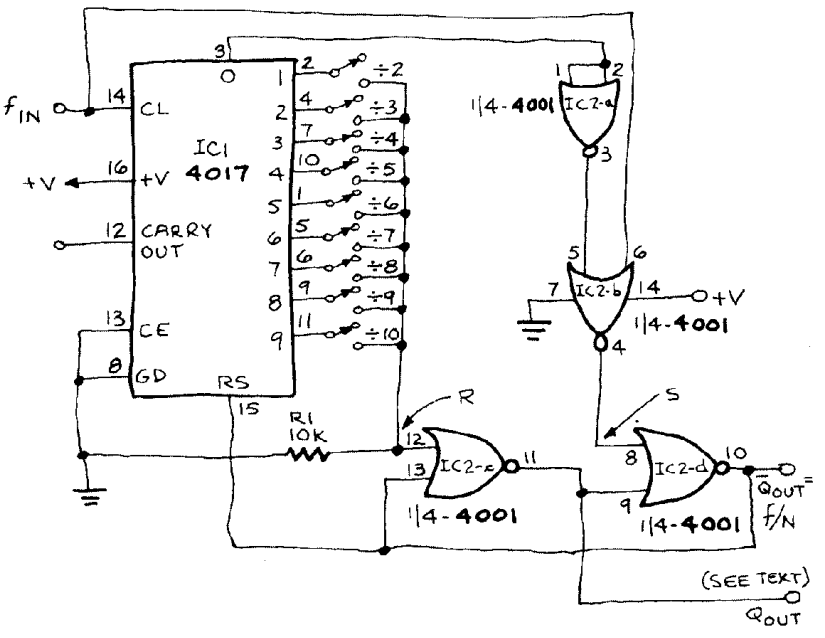

FIG. 3

continued from page 88

inputs are grounded. Since the $\overline{Q}$ output is also connected to one of the legs of the first NAND gate we have two highs there and the Q output will be low. If we switch the S input to +V, the output of the second NAND gate won't change because the other leg is still being held low. Suppose we now connect the R input to ground while the S input is switched to +V. With one leg grounded, the Q output will go high and the two highs at the inputs of the second NAND gate will make the $\overline{Q}$ output low. As you could predict, that flip-flop only responds to negative triggering because of the basic operation of the NAND gate.

The only thing to watch out for when you're using this flip-flop is to make sure that the S and R inputs aren't grounded at the same time. If that does happen, both outputs will be high and the circuit will be unstable. In practice, the last input to be grounded will decide the ultimate state of the flip-flop. But don't believe us—build it and try it yourself.

We can build the same sort of circuit with NOR gates and the flip-flop will respond to positive triggering. Use the previous discussion as a guide and trace through the operation of the NOR gate flip-flop so you understand how it works.

Now let's get back to our original problem. The circuit in Fig. 3 uses a NOR gate flip-flop to control the operation of the reset pin on the 4017. We're using NOR gates because they respond to positive triggering and the outputs of the 4017 are active high. The truth table for the flip-flop is shown in Table 1. The not-allowed state with NOR gates is having both the flip-flop inputs high. This isn't a problem, since the internal gating of the 4017 guarantees that only one output can be high at any one time.

As long as none of the switches are closed, R1 holds the input of the flip-flop low. That means that the $\overline{Q}$ output will be low regardless of what is happening at the S input. The reset pin of the 4017 is also held low and the chip is enabled. Now let's close one of the switches and see how the circuit works.

When the selected output goes high, the R input of the flip-flop goes high and a high appears at the $\overline{Q}$ output. This resets the 4017, the selected output goes low, and the 0 output, pin 3, goes high. Remember that the 4017 outputs go high in turn and whatever output you select will be low immediately following a reset pulse. That makes the R input low and control of the flip-flop moves over to the S input. You can see from the truth table in Table 1 that we need a positive pulse there to make the flip-flop change state and put a low at the $\overline{Q}$ output to release the 4017's RESET pin.

The Ø output of the 4017 is inverted by NOR gate IC2-a and presents a low to one leg of NOR gate IC2-b. The other leg of the gate is connected to the input clock and when a low appears there, IC2-b goes high and resets the flip-flop. That releases the RESET pin and enables the 4017. If you keep the switch closed at the keyboard, the circuit will reset over and over at the same point. The result will be a series of pulses at the $\overline{Q}$ output equal to the input frequency divided by whatever number you chose.

That circuit gives the 4017 a reset operation that is both synchronous and locked to the input clock. By following everything carefully you should have no trouble understanding how we did it. Remember that the reset operation starts on the positive half of the incoming clock cycle and is ended on the negative half of the same clock cycle. Since the input clock will be running faster than the pulses at any of the 4017 outputs, we don't have to worry about glitching in the count.

A side advantage of that approach is that the Q output of the flip-flop will give us an output wave that is equal in frequency to the $\overline{Q}$ output but opposite in sign. That can come in handy for some things and is especially nice since we're getting it for free.

If you want to cascade several 4017's together to increase the range of division, you won't be able to use the carry output, pin 12. Since that pin is high for the first half of the 4017's full count and low for the second half, frequency division of less than six will mean that the carry pin never goes low. The 0 output will, however, always go through a full cycle no matter what division you're doing, so you can take your signal from there.

The duty cycle problem

Now that we've solved the reset problem, let's look at the duty-cycle problem. In case you forgot what it is, we discovered that the duty cycle of the output would change every time we divided by a different number. It would follow the form 1/N where N is the number you're dividing by. More specifically, the high time would be equal to the period of the incoming clock, and the low time would be equal to N − 1 times the period.

If you're dividing by an even number, some simple gating would let you get an output with a nice 50% duty cycle but trying to do the same thing with an odd number would be—well, odd.

One of the basic rules of design is that there's a better way to do everything and that's true here. When simple problems generate overly complex solutions, it's time to scrap your whole approach and start over with a different color paper. In this case, squaring up the duty cycle not only calls for a different approach, it calls for a different IC—a different kind of counter. We'll examine that—and other mysteries—in next month's column. **R-E**