Frequency Division With Shift Registers

A simple approach to dividing by even and odd numbers while maintaining 50% duty cycle

By W.L. Hoffman

Over the year, many schemes have been developed for dividing frequencies by such odd numbers as 3, 5 and 7. The basic division isn't much of a problem since this can be obtained in a number of ways. The problem is in obtaining a symmertically square wave with a 50% duty cycle after division. This article explores the use of shift registers and exclusive-or (XOR) gates as both even- and odd-number frequency dividers that do just this.

How Shift Registers Divide Frequencies

Figure 1 illustrates a generic type serial-in/parallel-out (SIPO) shift register. This type of shift register (hereafter referred to by its initials "SR") accepts binary data one bit at a time at its input and "shifts" it one position down the line of its Q outputs for each rising clock-pulse edge.

For example, if a binary 1 is applied to the input, the 1 will appear at the Q1 output following the rising edge of the next clock pulse. After the rising edge of the second clock pulse, the 1 shifts to output Q2 and output Q1 receives new data from the input. For each successive input clock pulse, data moves down the Q output line one position, with data applied to the input shifted to the Q1 output each time.

To make the SR into a frequency divider, you connect an inverter from one of the Q outputs to the input terminal, as shown in Fig. 2. Clearing the SR before applying clock pulses sets all Q outputs to 0, with a binary 1 fed back to the input. For the first three clock pulses, 1s will shift into the SR making Q1, Q2 and Q3 all 1s, with 0 at the input. The fourth, fifth and sixth clock pulses shift a string of 0s into the SR.

As the clock continues, alternating strings of three 1s and three 0s move down the register and appear at the output as a square wave with a frequency of one-sixth the clock frequency. If the inverter had been connected at Q4, the clock frequency would have been divided by 8, while connecting it at Q5 would divide by 10, and so forth. So far, you've seen

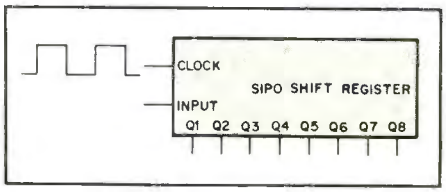

Fig. 2. Shift register frequency division occurs when the input is its own inverted output.

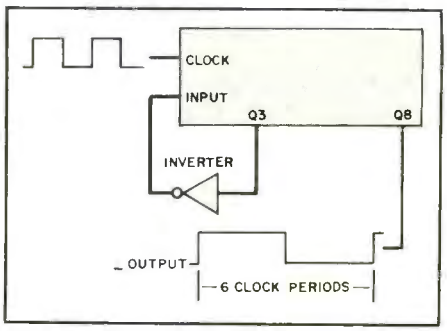

Fig. 1. A typical serial-in/parallel-out (SIPO) shift register.

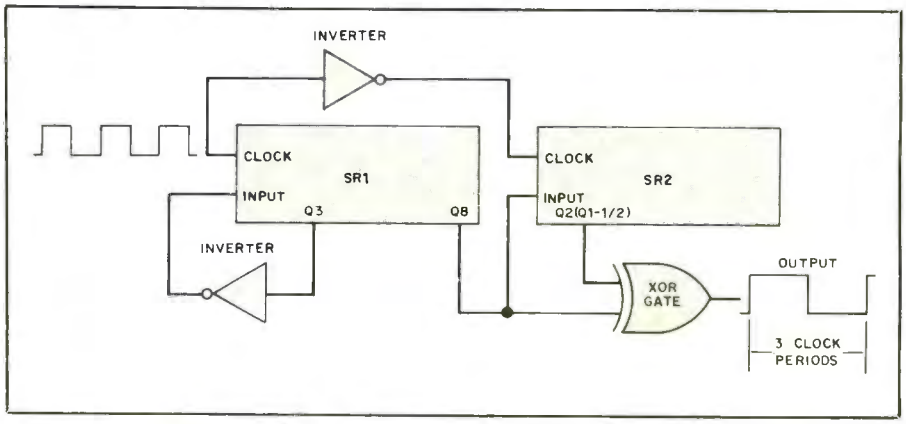

Fig. 3. The inverted clock makes the output of the second shift register equivalent to half positions of the first shift register's outputs.

how a single SR can work as a frequency divider for even numbers that are equal to twice the Q position used for feedback.

Dividing by Odd Numbers

If you want to divide by an odd number, you might think that you need an SR with half positions! You don't. By adding another of the same type SR and inverter arrangement as used in Fig. 2, you can obtain division by odd numbers. Such an arrangement is shown in Fig. 3. Here the output at Q1 of SR2 is identical to the SR1 output but is delayed by half a clock period because SR2 is driven by the inverted SR1 clock. This causes the positions on SR2 to be equivalent to half positions on SR1. Therefore, you can consider the SR2 positions as Q½, Q1½, Q2½, etc. instead of Q1, Q2, Q3 . . .

Notice on the timing diagram of Figure 4 that the waveform from Q8 of SR1 differs by 1½ clock periods from the waveform of Q2 of SR2. The XOR of these two waveforms is the third line of Fig. 4 and has the desired divide-by-3 frequency of the clock with a 50% duty cycle.

To divide by other odd numbers, you can move the feedback connection to the corresponding Q number on SR1 and change the XOR input to the appropriate half position of SR2. For example, to divide by 5, move the feedback to Q5 of SR1 and the XOR input to Q3 (Q2½) of SR2.

To divide by even numbers, remove the XOR input from SR2 and tie it to the + V supply rail. Place the feedback position at an output number on SR1 that is half the divide-by number.

Figure 5 is a complete schematic of a circuit built around shift registers and XOR gates for dividing by odd numbers up to 15 and even numbers up to 32. Notice that XOR gates replace the inverters and an additional SR follows SR1 to extend the division range. **ME**

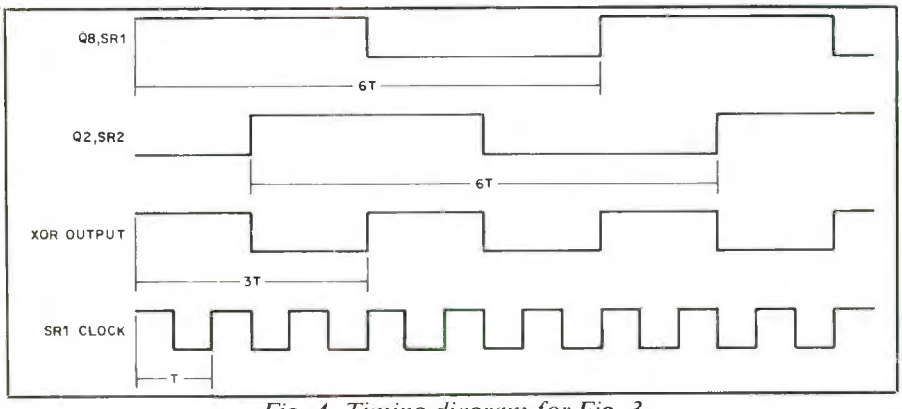

Fig. 4. Timing diagram for Fig. 3.

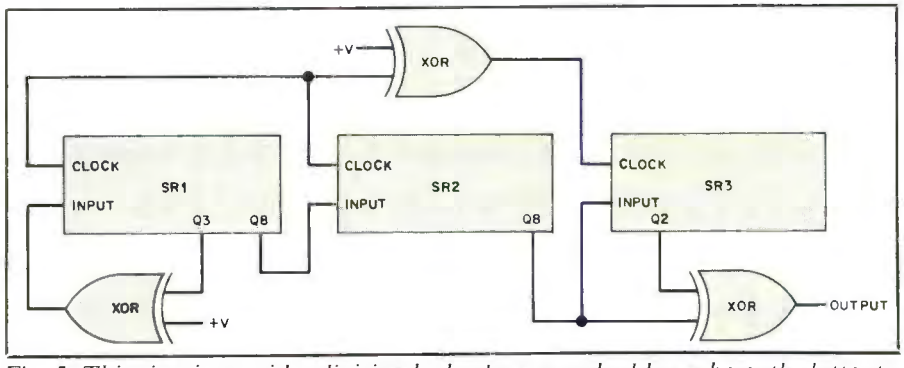

Fig. 5. This circuit provides division by both even and odd numbers, the latter to a maximum of 15. The arrangement shown divides by 3.