Simple square-wave generator, hardware digital divide-by-seven circuit

By Don Lancaster

Let's start off this month's column with some updates. Apparently *Airborne Sales* went belly up, so your best source of three-way EGR valves remains *Jerryco*. The best and cheapest source I've found so far for low-pressure pneumatic robotics air connectors is *Value Plastics*.

There's at least one major manufacturer of stock project cases that have built-in battery compartments. Check out *Pac Tec* on their HPS-9VB, HPL-9VB, and HP-BAT-9V cases, all of which are available in blue, tan and grey. Most cost less than ten bucks.

As usual, all the names and numbers are at the end of this column. By the way, some of these sources are *very* hard to find on your own, so you might want to start your own names and numbers notebook. I subscribe to over 200 technical magazines and scan quite a few more to bring these sources to you.

Also as usual, keep those calls and letters coming. I'll be happy to send some free *Applewriter* patches to anyone who asks for one.

I may be gone next month on a world class tinaja quest, so I've asked Marcia Swampfelder to make one of her rare guest appearances. Don't miss it.

I need a hardware digital divide-by-seven.

This is a tad heavy and specialized for this column, but since it shows us some important fundamentals, let's have at it.

Say you want some oddball math function or something else that is complicated but not totally irrational. How do you do it? First, you must ask whether you want to use hardware or software. Software, once developed, is cheap and easy to change for upgrades or customization. It is infinitely duplicated with little or no cost or on-hand inventory. Software, then, is usually far and away your best choice in this case.

On the other hand, hardware is usually much faster and is called for whenever operating speed is super important.

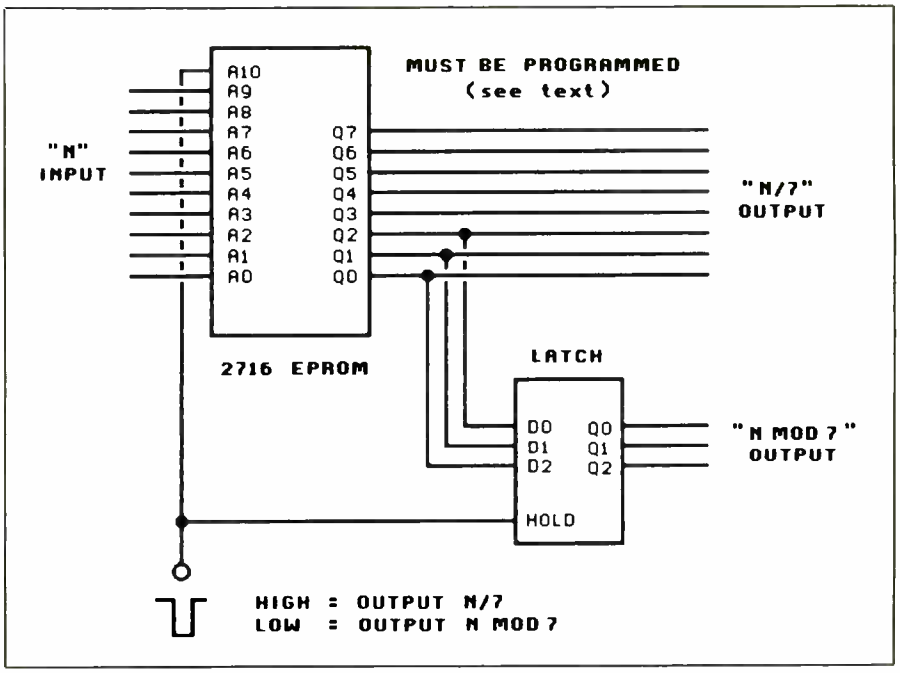

Fig. 1. This is one way to use an EPROM to do "table lookup" of a fancy math function. The scheme shown time-shares one EPROM to handle both the N/7 result and the N MOD 7 remainder.

Hardware is also needed if your circuit is so simple that it does not already have a CPU/RAM/ROM setup already inside it and ready to go, or if your CPU has better or more important things that you need done right now.

Second, you must ask whether you are going to look up the answer or calculate it. With the *table lookup* method, you simply get the answer out of an exhaustive list of all possible answers. Table lookup is very fast, but needs lots of memory or hardware storage space. Table lookup is usually the best hacker choice, provided the table can be kept reasonably small, say less than 4096 total entries. There are tricks to shorten tables, such as factoring, compression, precoding, interpolation, table pairs, symmetry, partial lookup, and so on.

With the *calculation* method, you actually calculate the result you are after, step by step, using some algorithm that does the job. The calculation method is always much slower, but often will need far less storage space or silicon real estate. Thus, you can look it up or calculate it with software, or you can look it up or calculate it with hardware. Usually, one of the four methods will be far better than the other three. Which one? That depends.

The most likely need for a divide-by-seven is to speed up Apple HIRES graphics animation. This is needed in calculating screen base addresses. A hardware divide-by-seven is the key to a 50× to 500× speedup of the best animation available at the present time.

The software solutions to this problem are well known, and appear in my *Enhancing your Apple II,* Volume I (SAMS 21822) and elsewhere in the Apple literature. The original software divide-by-seven was done by actually casting out sevens, painfully one at a time until none were left. The number of sevens cast out was the result, while anything left over was the modulo. While the code was incredibly compact (in those days, a 4K Apple was a really big machine), the result-

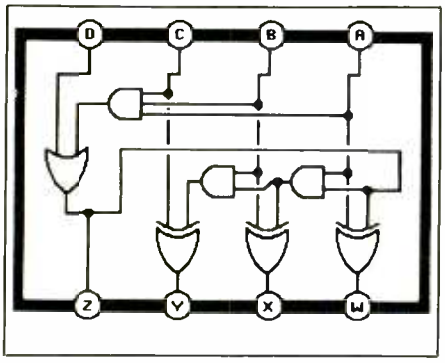

Fig. 2. Logic can be used to calculate a fancy math function. This module handles one step of a binary ÷ 7.

ing animation speed was only useful for such things as modeling earth tides, snail geriatrics, congressional reform, or glacial varves. A typical calculation time was 500 microseconds per screen base address.

Modern Apple arcade graphics instead use table lookup to handle the divide-by-seven. The HIRES screen base addresses are gotten out of a table in memory. While the table lookup only takes a few microseconds, a total of 50 microseconds per screen address is involved with the overhead. The tables take some 500 bytes or so of memory, less if you get sneaky.

By using hardware on a plug-in card, you should be able to reduce the divide-by-seven time and its overhead to one or two microseconds at most.

Let's look at two possibilities.

The table lookup can be handled by a pair of 2716 EPROMs. One handles the N/7 result, while the other handles N MOD 7 remainder. When properly programmed, you simply input your straight binary word, and your result or remainder pops out ready to use a few hundred nanoseconds later.

EPROMS work by exhaustively decoding all possible input combinations and then giving you a unique output word of your choice for each and every possible combination.

I have the source code available under EDASM for this N/7 and N MOD 7 EPROM pair. A printed copy is yours free on request, while both the ready-to-change source code and the ready-to burn hex object code costs $9.95 on disk from *Synergetics*.

Pairs of EPROMs are no big deal, but chances are you may need the space or the power consumption elsewhere, particularly if you cannot afford CMOS EPROMs. So Fig. 1 shows a sneaky trick that lets us *time-share* one EPROM for two different tasks.

Essentially, address A10 splits the 2716 EPROM in half. If it is high, you get the N/7 code. If it is low, you get N MOD 7. To separate the two, you catch the N MOD 7 output with a latch and hold it. You look up the N MOD 7 result only as long as you have to per the response time of the EPROM, typically 250 microseconds. Naturally, you look at and use both results only when they are both valid. While only half the speed of two seperate EPROMs, this can be more than fast enough for many needs.

Should you have fancier requirements, you can step up to a 2732, 2764, 27128, 27256, or even a 27512 EPROM. Other times, a second or third 2716 can be used for more than eight different outputs. Often though, it is far better to rethink and compact the stuff needing looking up down to a minimum size.

What about calculating a hardware divide-by-seven?

Almost always, table lookup is simpler and cheaper, particularly for a hacker's budget. A while back, I almost bought a gate array to simplify speedup of Apple graphics. In a gate array, the amount of silicon real estate is super critical, so I had to calculate a divide-by-seven. Let's see how it is done.

You can do a binary long division in pretty much the same way you did decimal long division in the third grade. See if the number will fit. If so, put down how many times it fits and then subtract to get what's left. If not, bring down another place and try again.

With a binary divide-by-seven long division, there are several possible simplifications. For one thing, your answer is always one or zero, meaning "yes it fits," or "no, its too big." Further, your subtraction is always a three-digit, or *octal* (ugh!) subtraction. But *substracting* binary *seven* in octal is the same as *adding one* in octal. And adding one with hardware is simpler that actually subtracting seven. Two's complement and all that.

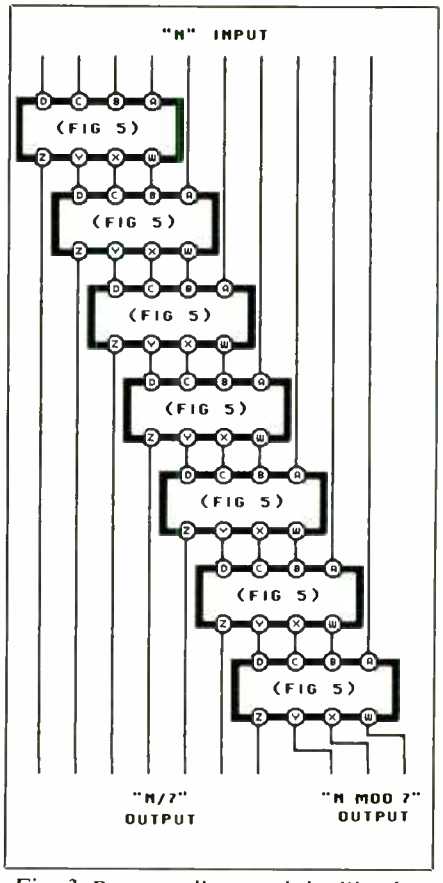

Fig. 3. By cascading modules like that shown in the previous figure, you can obtain a logic circuit that performs the complete ÷ 7 operation.

At any rate, Fig. 2 shows a module that handles one step of a binary divide-by-seven division. It obeys the rules:

IF D = 1 OR IF C + B + A = 7 THEN Z = 1
IF Z = 0 THEN Y + X + W = C + B + A
IF Z = 1 THEN Y + X + W = C + B + A + 1

Org! What a mess! Let's try it in English instead. Z is the result of the division of this stage. Z is one if what's left over from before is seven or more, since seven fits into seven or more. Z is a zero if

what's left over from before is six or less, since seven will not fit into anything smaller than itself. So much for the result.

If seven did not fit, you simply use the three lowest bits of what was left from before over again. Thus A becomes W, B becomes X, and C becomes Y. If seven did fit, you have to subtract seven from the three lowest bits of what was left from before. But, thanks to some hairy two's complement thinking, *subtracting* seven in octal is the same as *adding* one. So, if seven did fit, you add one to the three lowest bits of what was left from before. These become the highest three bits for the next step in the calculation.

Figure 3 shows how you cascade modules for a complete divide-by-seven. The Z output of each module forms part of the N/7 output. The Y, X, and W outputs become the three highest inputs for the next stage. The lowest, or A input of the next stage, drops directly down from the N input.

The result is an N/7 output at the left-bottom and the N MOD 7 remainder at the right-bottom. For more precision, you can add more modules.

Sharp-eyed readers may note a logic error on the first stage for N higher than decimal 959. In the intended use, the maximum possible input N is only decimal 558, and N values starting with four binary ones will not normally occur.

If this bothers you, another module can be added at the top with its D input grounded. Gate array people will go up the wall calculating the worst-case propagation delays on this. But, at worst, it should be faster than an EPROM.

Anyhoo, here is a hardware divide-by-seven. And, if you really wanted to do one this way, a fist full of CMOS jellybeans will do the trick. I count a dozen at fifty cents each. Can you do better?

In fact, let's see your "best" possible hardware or software divide-by-seven. Can you use those new PLA programmable logic arrays now that they are coming down in price? What new software tricks can be done to minimize calculation times or lookup storage area?

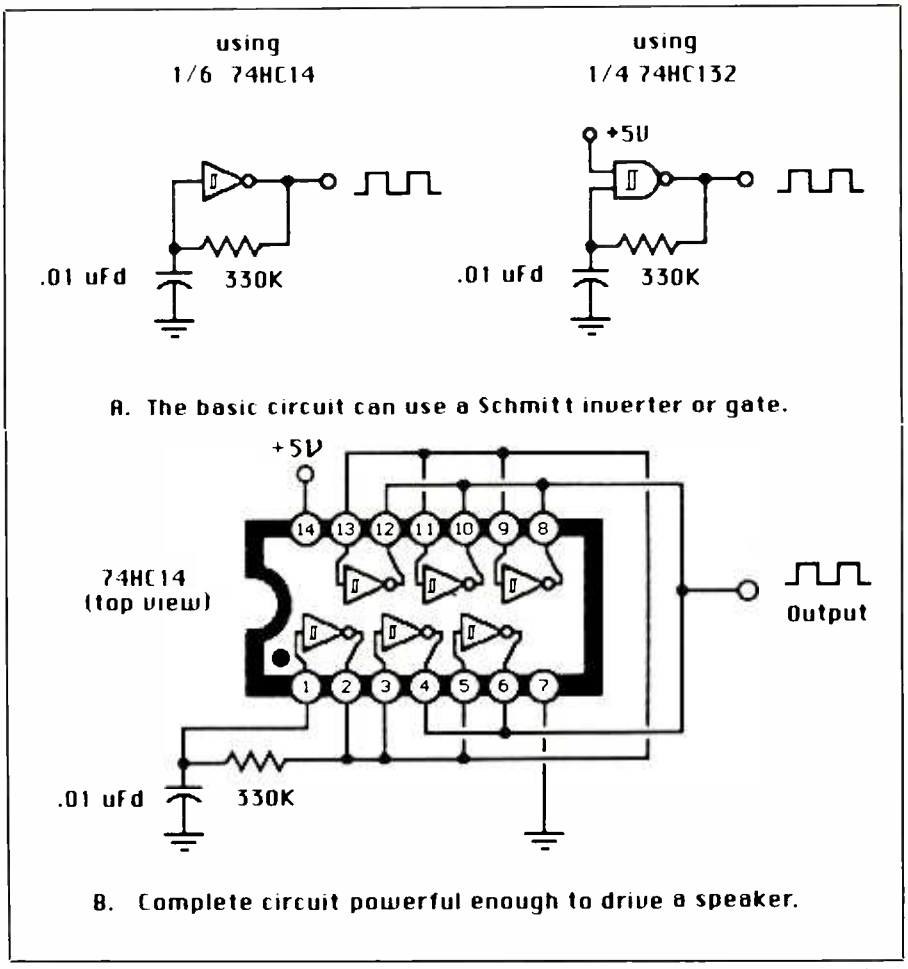

Fig. 4. Illustrated in these drawings are some square-wave signal generators in which are used CMOS Schmitt-trigger integrated circuit devices.

You can also combine the best parts of Figs. 1 and 3, using one trip through a 2716 EPROM and three simple gates. Do you see how?

Show me a simple square-wave generator.

This is one of my favorite circuit tricks, since it is both elegant and simple. It's also an ideal first project in electronics.

There are several different CMOS Schmitt triggers on the market. These include the 4093, 74C14N, and the newer 74HC14 and 74HC132. The usual sources are *Motorola, National,* and *Texas Instruments,* among dozens of others. Check the ads in the back of *Modern Electronics* for availability.

The 4093, 7314N, and 73HC14N are hex inverters, while the 4584 and 74HC132 are quad NAND gates. Cost is usually under a dollar.

Any of these will make a jim-dandy square-wave generator (Fig. 4). All it takes is one CMOS Schmitt inverter or Schmitt NAND gate, one resistor, and one capacitor.

Here's how it works: The inputs to all CMOS logic are essentially open circuits.

Thus, a CMOS gate or inverter does not significantly load whatever is driving it.

Further, with a CMOS Schmitt trigger, the logic has a built-in *hysteresis,* or snap action. This means that a positive-going waveform will not change the output until it is well over halfway to the positive supply voltage. A negative-going waveform will not change the output until it is well under halfway down to ground. Between the two is a *window* in which the output stays up if it was up and down if it was down.

This snap-action is normally used to clean up a noisy or slowly changing input signal. Thus, the intended use of Schmitt triggers is to detrash sloppy inputs at the front end of your circuitry. We will use this snap-action feature in a slightly different way.

Now, follow the bouncing ball. Assume we first power the circuit. The charge on the capacitor is zero and it cannot be instantly changed. Either the inverter or the NAND gate will have a high (or positive) output, since both of these invert their logic.

A high output slowly charges the capacitor through the high-value charging resistor. The CMOS input does not load the capacitor, so the capacitor starts building up a positive charge.

Eventually, the capacitor passes the lower trip point of the snap-action window, but nothing happens, since the output is already high. The capacitor keeps charging. When it gets to the upper trip point, the output snaps down. The output has now switched from a high state to a low state.

At the same time, the resistor now begins discharging the capacitor, since the capacitor is positive and the other end of the resistor is nearly at ground. The capacitor continues discharging only to the lower trip point, at which time the output goes high and the action repeats.

After the first cycle, the capacitor will saw itself between the upper and lower trip points. Typically you will see a 1-volt sawtooth wave across the capacitor, centered at half the supply voltage. The output will be a clean square wave of nearly perfect symmetry. The frequency is fairly independent of temperature and supply voltage as well.

Resistor values can be into the megohms, and capacitor values can be into the microfarads, letting you time out to minutes or more. Accuracy at extremely long time delays won't be all that great though, so go to the digital counter route if you need extreme accuracy. The highest frequency you can get with this approach is half a megahertz or so.

You can easily vary the frequency by substituting a potentiometer for the charging resistor. If you do this, be sure to add a series 10k resistor to keep the capacitor from shorting the output at the pot's minimum setting. For wide range changes, the capacitor may be switched in decade (10×) steps.

Here's an off-the-wall hint. If you put a dial on the panel behind the pot, the numbers on the dial will end up very cramped and nonlinear. The solution to this is to use an *audio*, or log-taper potentiometer, and then to put the dial on the *shaft* and the marker on the *panel*. The nonlinearity is caused by the frequency being inversely proportional to the pot setting. Reversing dial and marker avoids using an extremely hard-to-find reverse-log-taper pot.

Figure one also shows you a complete square-wave generator test instrument that is powerful enough to directly drive a speaker. All this does is use the remaining five inverters in parallel as an output driver. Total circuit cost is under $2.

The very first cycle on power up will be longer than the others, since the capacitor has to charge all the way from ground, rather than from only the lower trip point. One place this extra delay comes in handy is for an automatic repeat function on a keyboard.

Several gotchas. Do not use TTL or LSTTL Schmitt triggers as they will load the capacitor too heavily and will not work at low frequencies. Use the 74HC14 or 74HC132 only over a +2- to +6-volt supply range. The 4093, 4584, and 74C14 may be used over a +3- to +15-volt range and thus may be powered from a 9-volt transistor battery.

In addition, make sure that *all* unused CMOS inputs go somewhere, such as to ground or the positive supply. Fail to do this, and your package current will dramatically increase, and noise can foul up the works. **ME**

Names & Numbers

Apple Computer
20525 Mariani Ave.
Cupertino, CA 95014
(408) 996-1010

Jerryco
601 Linden Pl.
Evanston, IL 60202
(312) 475-8440

Motorola, Inc.
Box 20912
Phoenix, AZ 85018
(602) 244-6900

National Semiconductor Co.
2900 Semiconductor Dr.
Santa Clara, CA 95051
(408) 721-5000

Pac Tec
Enterprise & Executive
Philadelphia, PA 19153
(215) 365-8400

Howard W. Sams & Co., Inc.
4300 W. 62 St.
Indianpolis, IN 46206
(800) 428-SAMS

Synergetics
Box 809
Thatcher, AZ 85552
(602) 428-4073

Texas Instruments Inc.
Box 225012
Dallas, TX 75265
(214) 995-6611

Value Plastics
5137 S. College Ave.
Fort Collins, CO 80525
(303) 669-4351

Need Help?

Phone or write your hardware hacker questions and comments directly to:

Don Lancaster
Synergetics
Box 809
Thatcher, AZ 85552
(602) 428-4073