DRAWING BOARD

Our descrambler is almost finished.

ROBERT GROSSBLATT

Believe it or not, there's not a lot more to the design of the basic descrambling circuit than the subject matter we've covered over the past few months. But if you're really interested in video, there's much more for you to learn—the video-scrambling stuff we've been going through is just one small part of the subject. Video is a complex business, and once you get the hang of how signals can be manipulated, you'll find that there's no end to the kind of circuitry you can design. There's also a huge market for it as well. But back to the subject at hand.

When we first started work on the SSAVI descrambler, the block diagram probably looked quite complex. But if you look at Fig. 1, you'll see that most of the work has been done. The most difficult job left for us is to come up with the counter that's needed by the phase-locked loop. It's a job we have to do carefully because the success of the descrambler depends on how well we can maintain the stability of the horizontal-sync signal.

The phase-locked loop in our design is driven by a 504-kHz clock based on an R-C network. That frequency was chosen for two basic reasons: The first is that it's exactly 32 times the NTSC line frequency of 15,750 Hz, and that makes it easy for us to do the division necessary to produce horizontal sync pulses. The second, and somewhat more subtle reason, has to do with the period rather than the frequency.

A 504-kHz clock has a period of 1.98 microseconds. That's interesting because, by using five of the counts, we can get a pulse that has a width of 9.92 microseconds, which is close enough to the NTSC's established standard of 10.7 microseconds for a horizontal blanking pulse.

Before we go any further, I have to comment on my apparent disregard for precision. I've been using language and design techniques that are loaded with phrases like "close to," "pretty much the same as," "in the ballpark," and so on. Now, I'm as impressed with standards as anyone I know, but numbers have to work in the real world as well as on paper. You have to keep things like cost in mind when you're designing a circuit.

The NTSC standards for a video signal have a precision of several decimal points, and every broadcaster in the country spends a lot of time and effort making sure his signal adheres to that standard. The same is true of TV manufacturers, as well. The theory for standardization is terrific but, as is usually the case, what happens in reality is different. All you have to do to see what I mean is put a scope on the video signal received by your TV set; you'll find an overall similarity to the video standard, but you'll be amazed at how sloppy the signal really is. The average horizontal frequency might be 15,750 hertz, but there's considerable jitter from line to line in the signal. The same is true for burst, blanking, and all the other components in the line.

Despite these variations in the signal, the picture that shows up on your TV screen is apparently unaffected by them. The difference between studio-quality video (the stuff on the monitors in the broadcaster's control room) and received video (the stuff you see on your TV) is minimal.

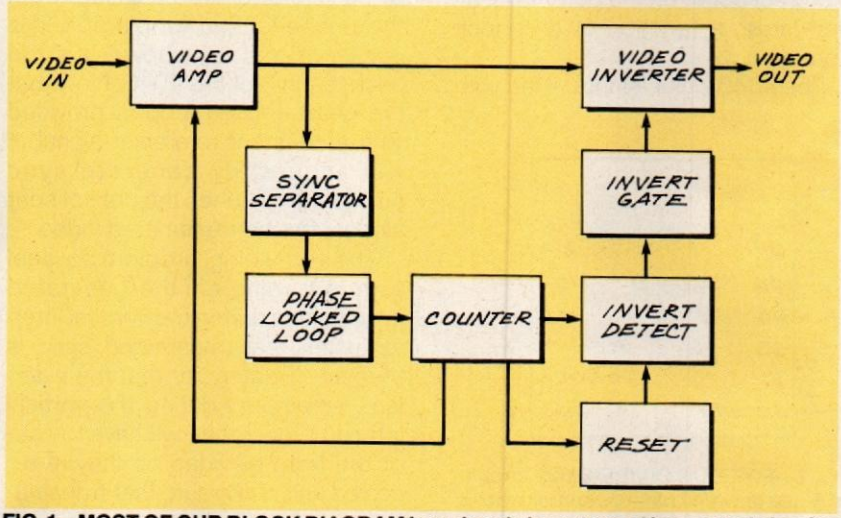

FIG. 1—MOST OF OUR BLOCK DIAGRAM has already been turned into actual circuitry. Now we need a counter for the phase-locked loop.

Keeping all that in mind, let's design the rest of the descrambler.

The next thing we need is a divide-by-32 counter to complete the phase-locked loop section of the circuit. After my discussion of the relative importance of precision, you should understand why the phase-locked loop's oscillator is only RC-based, and not crystal controlled: It's just cheaper and easier.

The divide-by-32 circuit (the "Counter" in Fig. 1) can be built in

several ways; as a matter of fact, we've designed a whole bunch of these things in the past. In order to keep the circuit as simple as possible, we'll use the 4040 binary counter whose pinouts are shown in Fig. 2. This is one of the earliest members of the CMOS family and still one of the best choices for general counting. It's a ripple counter, rather than a synchronous counter, so don't use it for applications where super accuracy is required. Remember that a ripple counter is a bunch of sequential counters, and each internal stage uses the output of the preceding stage as an input. This means that the outputs change in sequential order, and an incorrect count will be present briefly on the pins. Since the problem is caused by the propagation delay of each counter stage, the duration of the incorrect count on the outputs is, by and large, a function of the clock speed.

The 4040's clock input is fed with

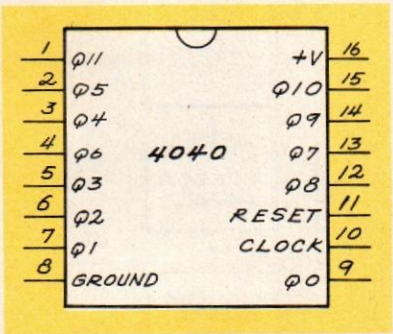

FIG. 2—FOR OUR DIVIDE-BY-32 circuit, we'll use the 4040 binary counter whose pinouts are shown here.

the 504-kHz signal generated by the 4046, and we're using a series of gates to decode the count and provide horizontal blanking and some other timing signals needed for the descrambler. The actual circuit is shown in Fig. 3. You should understand why each of these signals is needed before you start hooking everything up to the back of your TV set. To see what we need from the counter, let's use it to restore horizontal sync and then figure out what else we need to completely unscramble the incoming video.

Getting a horizontal sync clock for the phase-locked loop circuit is simple. All we have to do is pick off

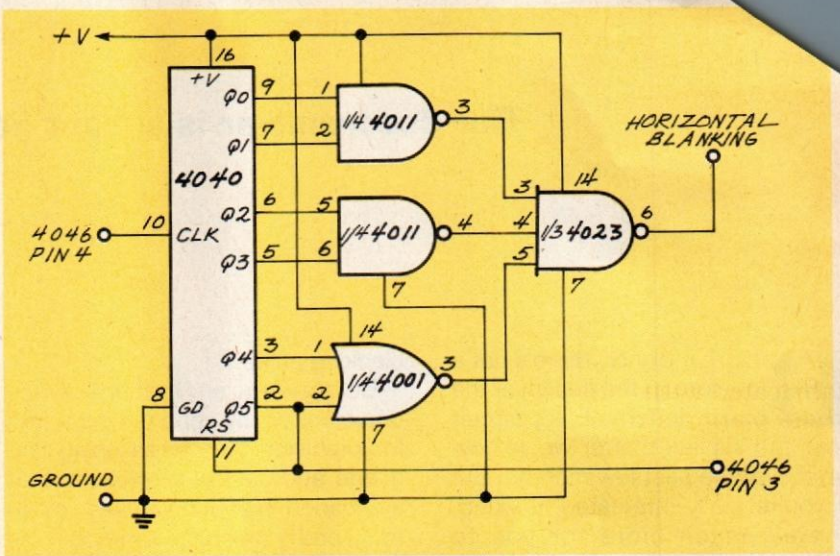

FIG. 3—THE 4040's CLOCK INPUT is fed with the 504-kHz signal generated by the 4046. We're using a series of gates to decode the count and provide horizontal blanking.

the Q4 output. That is a divided-by-32 version of the input clock from the phase-locked loop's 504-kHz oscillator and, as shown, it's fed back to pin 3 of the 4046. Now that the phase-locked loop is provided with a constant reference signal, it will accurately generate sync pulses—even when they're not sent along with the transmitted video.

The remaining problem to deal with is turning off the generated sync pulses during the vertical interval when real transmitted sync is present. (Remember that the video isn't scrambled during the vertical interval.) To do that, we have to count the lines of video as they're received and make sure that transmitted sync is processed for the first 26 lines of each frame. The starting point for the count is the vertical sync signal and, as you recall, we've already isolated the signal. The sync separator we built earlier produces a positive-going version of vertical sync. If we use the rising edge of that signal as the zero point for the counter, we have to count a number of lines to reach the point where the lines of video are carrying picture information and are therefore scrambled.

The two lines in the frame that mark the beginning and end of the transmitted horizontal sync are 260 and 27, respectively. We need a circuit that can count the received lines and let the rest of the descrambler know when to use received horizontal sync and when to use the artificial sync generated by the phase-locked loop. The zero point for the counter will be the rising edge of our vertical sync signal. Since that occurs at line 3, we have to decode a count of 24 and 257.

We can use a 4040 to do that job as well. This is the same sort of counter decoding we've done before in this project and others. When the counter reaches 24 we have to enable a gate that will send the phony horizontal sync pulses to the video amplifier at the circuit's input. When we get to a count of 257, the gate has to be disabled to let the real horizontal pulses through to resync the phase-locked loop circuit.

Decoding those two numbers is a pain in the neck since it involves watching nine counter lines. You can do that with a bunch of gates, but a better way is to use an EPROM. The choice is yours although, if you haven't had much experience in this sort of decoding, it's probably a good idea to work out the logic and build the decoder with gates. Otherwise an EPROM is the way to go. Assuming, of course, that you have programming capabilities.

When we get together next time, I'll give you the truth table for the EPROM and we'll finish off the descrambler. All that's left is a couple of flip-flops, a bit of work on the video section, and a way to detect the presence of normal or inverted video. Simple stuff! Ω