The Versatile NAND Gate

By Forrest M. Mims III

A sign over my desk reads, "Give me a bucket full of NAND gates, and I'll build you a computer!" The sign could be altered to specify NOR gates. In either event, a completely functional digital controller or simple computer can indeed be designed using only NAND- or NOR-gate building blocks.

Of course, the rich diversity of reasonably priced integrated circuits available to electronic circuit designers and experimenters precludes the necessity of wiring together dozens or even hundreds of gates to accomplish a complex function. Nevertheless, it's easy to overlook the versatility of simple logic gates.

I was recently reminded about the importance of simple logic gates while writing *Engineer's Mini-Notebook: Digital Logic Circuits*, a new Radio Shack book that should be available shortly after this column appears. Though this book describes applications for only a handful of chips, it includes nearly 100 application and interface logic circuits, and dozens more could have been added had there been space.

To demonstrate the versatility of simple gates, this entire column is devoted to one of the most basic of gate packages, the 7400 quad NAND gate. With nothing more than some 7400s, a few resistors and LEDs, a few dozen jumper wires, and a solderless breadboard to plug everything into, dozens of different logic circuits can be assembled. Spending a few hours designing logic circuits composed of only NAND (or NOR) gates can provide a valuable review of the basics of digital logic for both beginners and experienced circuit designers. At the very least, the experience can provide a refreshing change of pace from the hyperactive world of advanced logic circuits like microprocessors, programmable logic arrays, and the like.

NAND Gate Basics

Figure 1 shows a NAND gate formed by ANDing (connecting in series) the collector-emitter conducting paths of two npn transistors. The circuit would form an AND gate if the output were taken from a load resistor connected from *Q2*'s emitter to ground. When the voltages applied to the input are at or near 6 volts (high) or at or near ground (low), the circuit will operate according to the truth table that accompanies Fig. 1.

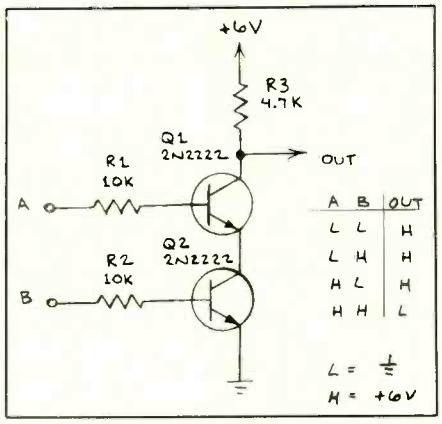


Fig. 1. A 2-input NAND gate configured using two transistors.

Occasionally, it may be advantageous to assemble NAND and other gates from discrete components. Generally it's much more convenient to use standard integrated-circuit gate packages such as the 7400, whose pin outline is shown in Fig. 2. It is this gate package which is used for all the circuits that follow. Though the 7400 is a TTL chip, it's possible to use functional equivalents. The 74LS00, for example, is a direct substitute that uses only about a fifth the current of the standard TTL 7400. Even less current is consumed by the 74C00. However, since the 74C00 is a CMOS chip, special handling and operating requirements must be observed.

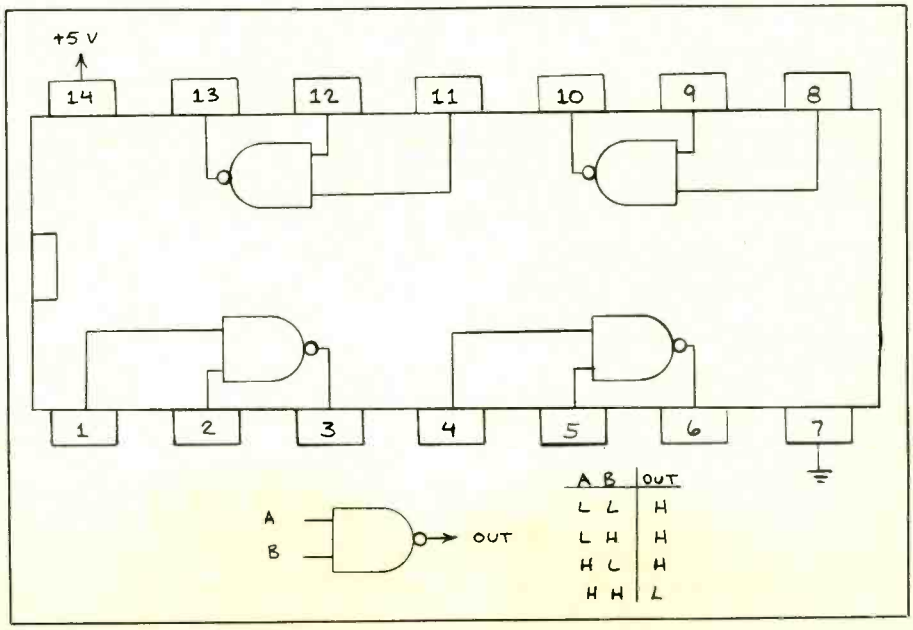
Figure 3 shows how four of the most basic logic functions can be achieved by interconnecting NAND gates. The key to assembly of these gates is the inverter, a function that can be achieved with a single transistor or by shorting together the inputs of a NAND gate.

TTL Logic Probes

Evaluating and troubleshooting logic circuits is greatly simplified with the help of a logic probe. Commercial logic probes are relatively inexpensive and usually include a pulse-trapping circuit for detecting and storing very brief pulses.

When a commercial logic probe isn't available or is in use elsewhere in a circuit, you can quickly assemble a pair of temporary probes from a single 7400 (or

Fig. 2. A 7400/74LS00 quad NAND gate.



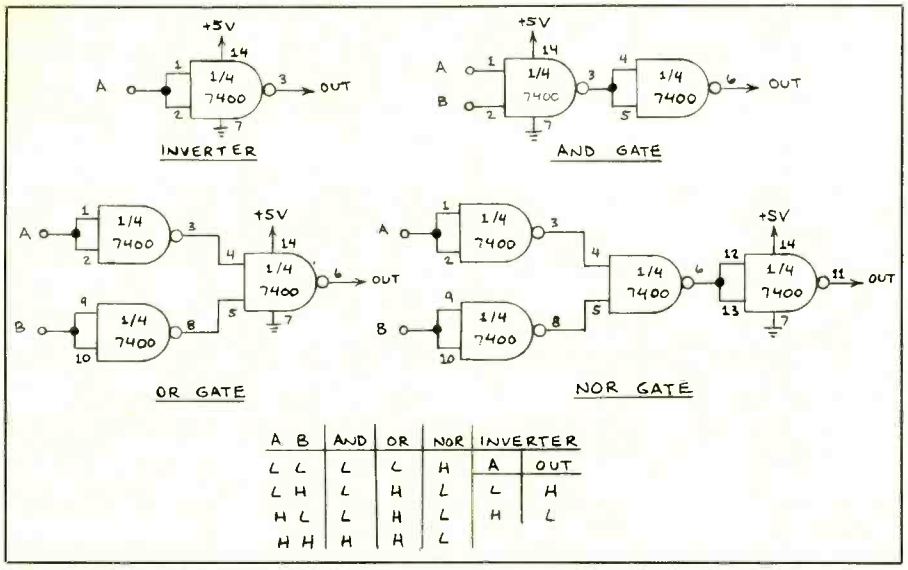


Fig. 3. Using NAND gates to assemble basic logic gates.

three probes from a 7404 hex inverter). Figure 4 shows two possible circuits, both of which use two inverters. The upper circuit displays the logic status at its input by means of a bicolor (red/green) LED. Once you learn to associate the two colors with their respective logic states, this single indicator circuit is very convenient to use. The lower circuit in Fig. 4 resembles conventional logic probes, since it uses a pair of separate LEDs to indicate the two logic states.

Incidentally, if you are building a fairly complex logic system, it's handy to have half dozen or more temporary logic probes like those in Fig. 4 ready to use at a moment's notice. I often dedicate a corner of a solderless breadboard or a spare breadboard for as many as a dozen temporary logic probes. They can be quickly connected to any part of the circuit for testing and be removed when they are no longer needed.

Combinational Logic with NAND Gates

Combinational logic circuits respond to incoming logic levels without regard to previous conditions. In other words, combinational logic includes no storage elements. The OR and NOR gates in Fig. 3 are examples of elementary combinational logic networks.

Figure 5 shows a rudimentary 4-bit decoder, a combinational network that can be easily modified to decode any combination of four bits. The output of the version in Fig. 5 remains high for all input combinations except one. When all four inputs are high, the output goes low. Therefore, this circuit decodes an HHHH input nibble. (A nibble is half an 8-bit byte.)

Figure 6 shows how the basic decoder in Fig. 5 can be expanded to decode other input nibbles. In Fig. 6, inverters have been inserted at the B and C inputs. Now the decoder ignores all but an HLLH input. For all other input combinations the output is high. For the HLLH nibble, the output is low.

No complicated procedures are required to custom-design any other 4-bit decoder. The object is for the output to go low in response to the appropriate nibble. We already know the output is low when all four inputs of the basic decoder in Fig. 5 are high. Therefore, all that's necessary to select a different input combination for decoding is to place an inverter at each low input. Therefore, as in Fig. 6, only the selected input nibble will supply an HHHH input to the basic decoder in Fig. 5. For example, to decode 0000, an inverter must be added to all four inputs to the basic decoder in Fig. 5.

Fig. 4. Simple TTL logic probes.

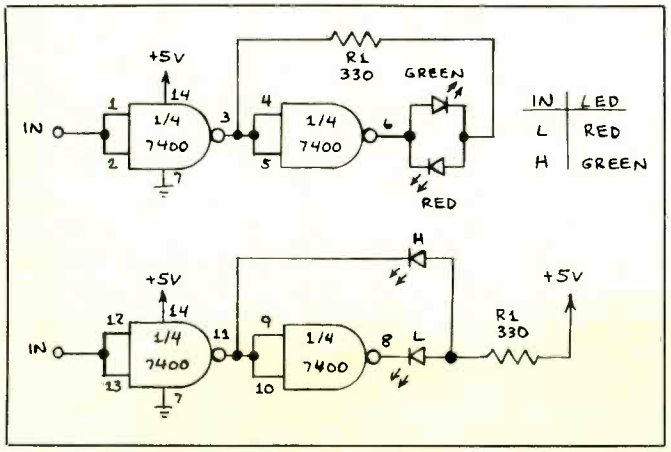
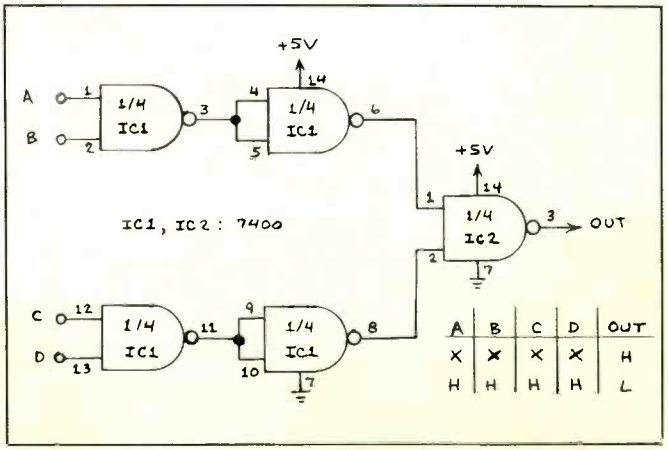


Fig. 5. A basic 4-bit decoder.



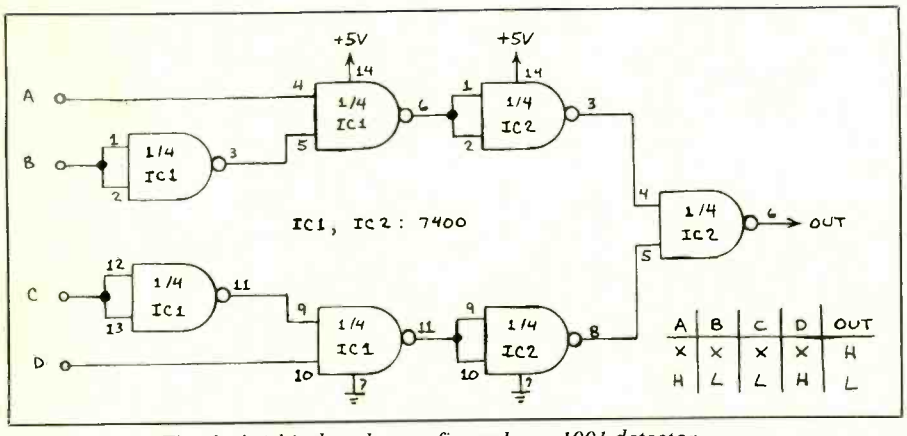


Fig. 6. A 4-bit decoder configured as a 1001 detector.

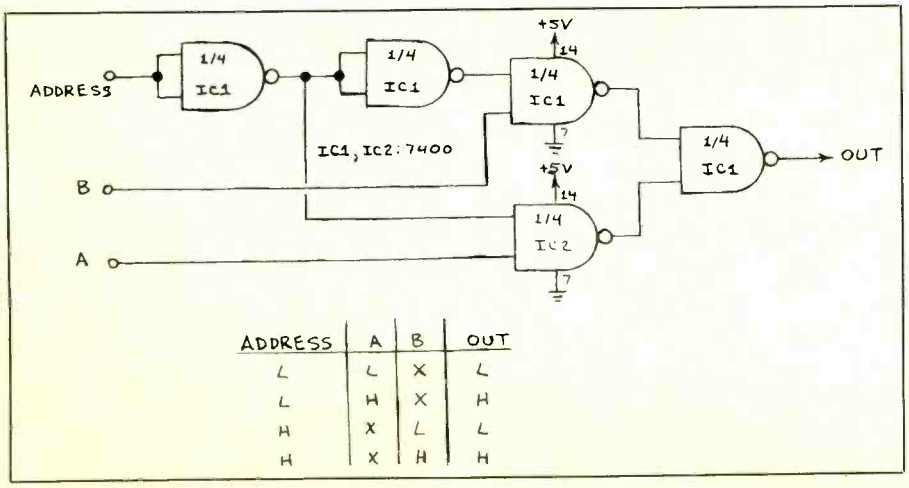
Now that you know how easy it is to design a simple 4-bit decoder, you will probably think of applications for these important logic tools. That's because decoders have an incredible variety of uses as stand-alone circuits. For example, the inputs to a decoder can be connected to various go/no-go sensors on an engine. If the predetermined combination of highs and lows from the sensors is not met, the decoder can actuate a warning light or shut off the engine. In combination with an input buffer, such as a shift register, a decoder can indicate the presence of a predetermined sequence of low and high pulses.

In a microprocessor, decoders can be used to respond to the bit patterns known as instructions. For instance, an 8-bit instruction word might be divided into two fields of 4 bits each. Each field is directed to a 4-line to 2-line decoder. Various patterns of bits in the two fields cause the decoders to enable and disable the control inputs of various registers and buffers tied to a common 3-state bus, thereby permitting the transfer of data in both directions through the bus.

Figure 7 shows how to assemble another important combinational circuit, a 2-input multiplexer or data selector, from nothing but NAND gates. Data selectors are the logical equivalent of mechanical rotary switches. The one in Fig. 7 is the equivalent of a 2-position switch. The address bit selects one of the two inputs and steers it to the single output.

Practical data selectors usually include four or more inputs and the required number of address lines. For example, a 16-line or 1-line data selector requires 4 address lines (four bits allows counting from 0000 to 1111 which is equivalent to 0 to 15, or 16 separate states).

Fig. 7. A two-input data selector.

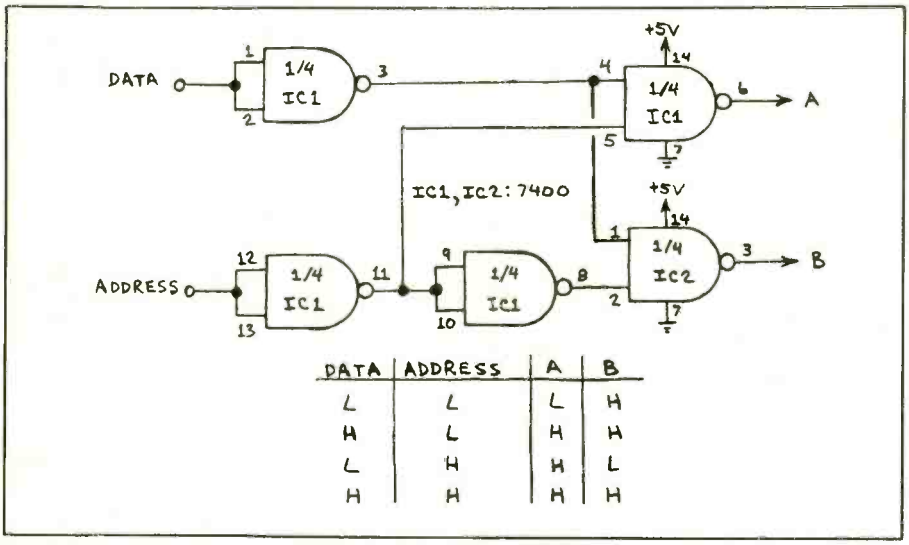


Data selectors can be used in standard combinational circuits or in sequential circuits in which the address bits are continually cycled by an external counter so that the input bits can be continuously sampled in sequence. A simple application for this operating mode is a bit-pattern generator. The pattern is designed by applying appropriate logic levels to the input lines. The speed of the counter determines the rate at which the bit pattern is cycled out the output. Another application is a parallel-to-serial converter. Here, a data word is applied to the inputs. The word is sent to the output bit by bit as the address inputs are incremented by a counter. The counter can also generate a control signal when all bits have been transmitted so that a new word can be entered at the inputs.

Figure 8 shows a 1-of-2 demultiplexer, the inverse of the data selector in Fig. 7. Here a single input bit is steered to the output port designated by the bit at the address input. An obvious use for the demultiplexer is to transform serial data back to its original parallel format.

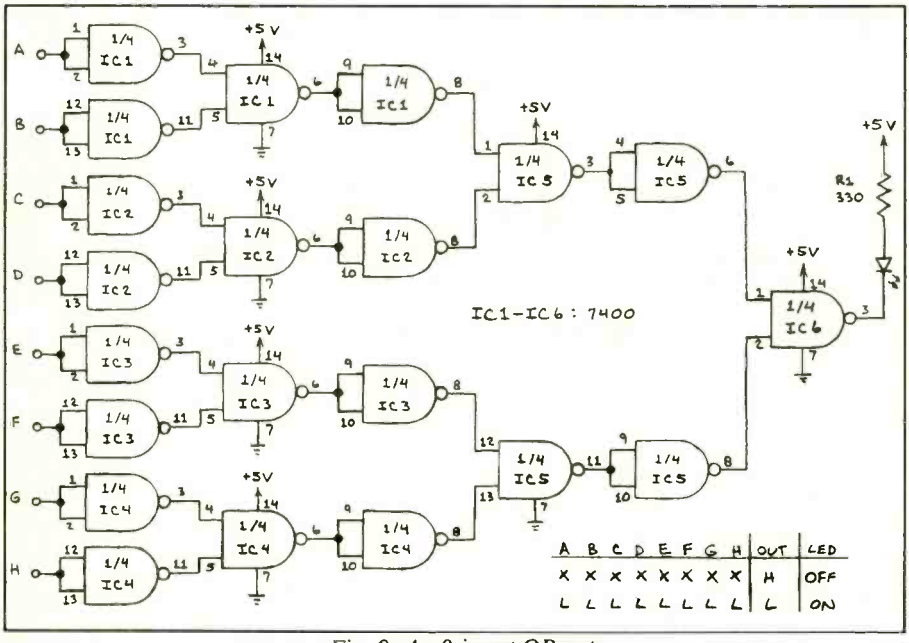
Consider, for example, a 2-bit communications link in which the output of the data selector/multiplexer in Fig. 7 is connected to the input of the demultiplexer in Fig. 8. A second line carries clock pulses to the address inputs of both circuits. The clock pulses simulate a 1-bit (0 . . . 1 . . . 0 . . . 1 . . .) counter. The bits at inputs A and B (Fig. 7) will be transmitted in serial fashion through a single line and reassembled into the original 2-bit pattern at the demultiplexer's output (Fig. 8). Rarely are only 2-bit patterns transmitted in this fashion. But the same principles can be applied to the transmission of much longer bit patterns.

Figure 9 shows an 8-input OR gate formed by a network of NAND gates. Only when all eight inputs are low is the output low. For all other input combinations the output is high. DeMorgan's theorem shows that a positive-logic (0 = L and 1 = H) OR gate is functionally



DATA	ADDRESS	A	B
L	L	L	H
H	L	H	H
L	H	H	L
H	H	H	H

Fig. 8. A 1-of-2 demultiplexer.



A	B	C	D	E	F	G	H	OUT	LED
X	X	X	X	X	X	X	X	H	OFF
L	L	L	L	L	L	L	L	L	ON

Fig. 9. An 8-input OR gate.

equivalent to a negative-logic (0 = H and 1 = L) AND gate. The circuit in Fig. 9 demonstrates this. Assume all eight inputs can be switched to +5 volts or ground by means of a row of toggle switches. If the switches are inverted (negative logic) so that their "on" positions indicate the eight gate inputs are connected to ground, then the circuit functions as an 8-input AND gate. Only when all eight input switches are "on" does the LED turn on. In this mode the circuit can be considered a unanimous vote detector.

Sequential Logic with NAND Gates

Sequential logic circuits are those in which the present state of an output is dependent on a previous operation. Generally, a clocking pulse is employed to trig-

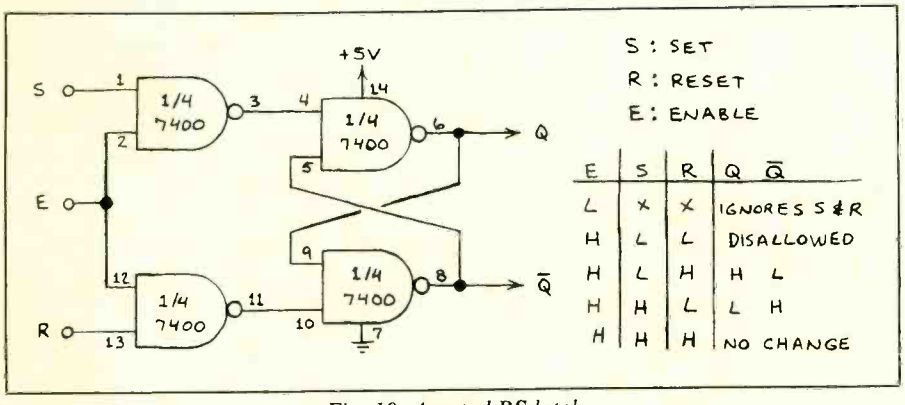


Fig. 10. A gated RS latch.

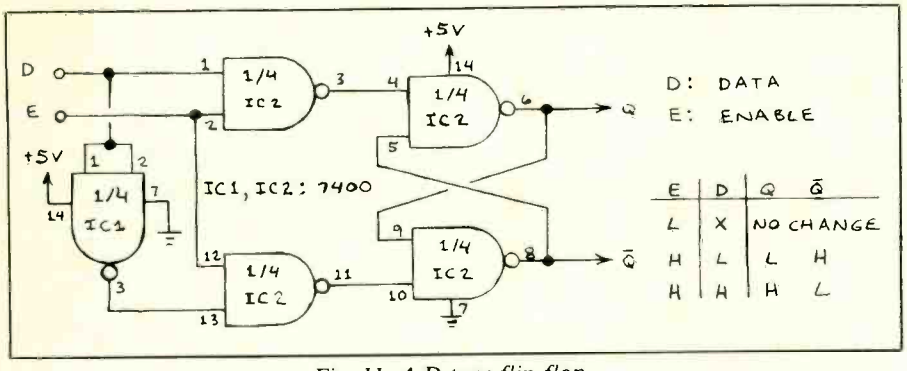


Fig. 11. A D-type flip-flop.

ger logic state changes in sequential circuits in a synchronized fashion.

Figure 10 shows a simple sequential circuit called an RS flip-flop or simply a latch. Note that this entire circuit can be made from a single 7400. The circuit ignores data at its S (set) and R (reset) inputs when the ENABLE input is low. When the ENABLE input is high, the latch responds as shown in the truth table. Latches can be used to store status bits. By extension, arrays of latches are used as memory registers.

As shown in Fig. 11, a D-type flip-flop can be formed from only five NAND gates. As with the latch, the D-type flip-flop ignores the data at its D input when the ENABLE input is low. Otherwise, the Q output of the circuit follows (stays equivalent to) the bit at the D input.

Figure 12 shows a circuit that is indispensible when designing and testing sequential logic circuits. It's a switch debouncer formed by cross-coupling the outputs with one of the inputs of a pair of NAND gates. The logic state of the single output follows the position of *S1*. The significance of the circuit is that it ignores the mechanical "bouncing" and consequent multiple output pulses that occur during closing and opening a switch. Instead, the circuit switches states on the arrival of only the first pulses in a series of bounced pulses, thereby ignoring subsequent bounces.

Bounceless switch circuits are handy for single-stepping a sequential logic circuit through its paces and observing what happens. Often, the only practical alternative is to slow the clock down to a few cycles a second and follow the action with logic probe(s), but sometimes that's not practical or possible.

Finally, Fig. 13 shows how to form a self-switching circuit by cross-coupling the inputs and outputs of a pair of inverters. With the values shown for *C1* and *C2*, the LEDs will flash on and off about twice each second. The value of capacitors *C1* and *C2* can be increased to slow down the flash rate. The circuit can also be used as a logic clock.

Fig. 12. A two-gate switch debouncer.

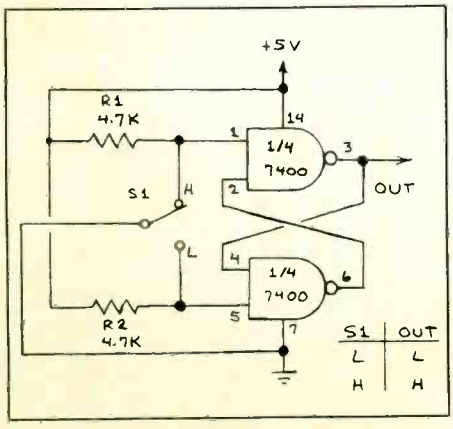
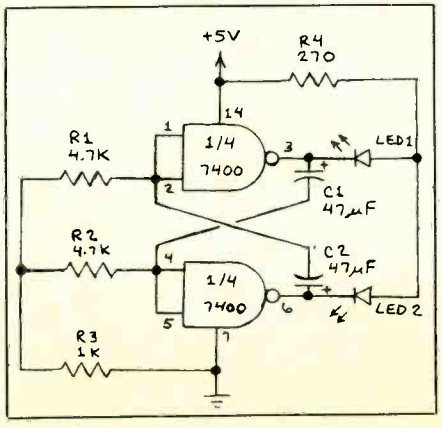


Fig. 13. Two-gate LED alternate flasher.



Going Further

The circuits presented here illustrate the role of the NAND gate as a fundamental digital logic building block. The NOR gate can also be used to accomplish these same functions. If you want to brush up on digital logic fundamentals, review the opening chapters in Don Lancaster's *TTL Cookbook* and *CMOS Cookbook*. Don's discussion of data-selector logic in these two classics is particularly good.

If you want to experiment with CMOS versions of the circuits presented here and others as well, see Radio Shack's *Engineer's Mini-Notebook: Digital Logic Circuits*. This book also includes numerous input, output and interfacing circuits for both the TTL and CMOS logic families of devices. **ME**