VOLUME 13, No. 4

Popular Electronics®

WORLD'S LARGEST-SELLING ELECTRONICS MAGAZINE

APRIL 1978/$1

How to Design Power Supplies
Easy-to-build LED Projects
Microprocessor Microcourse, Part II

16-PAGE CB RADIO SUPPLEMENT, INCLUDING
Buyers Guide to Super CB Mobile Transceivers

Issue

Sharp RT-3388 Computer-controlled Cassette Deck

Wharfedale E-50 High-efficiency Speaker System

Harman-Kardon Model 730 AM/Stereo FM Receiver

BY FORREST M. MIMS

Micro-PROCESSOR MICROCOURSE

PART 2. BASIC DIGITAL LOGIC

The Basic Logic Gates. All digital logic circuits, from the simplest counter to the most sophisticated microprocessor, are made from interconnected combinations of simple building-block circuits called logic gates. There are four basic gates, and they are designated according to their function as YES, NOT, AND, and OR circuits. Each of these basic gates has one or more inputs, a single output, and a couple of power supply terminals.

Various combinations of the binary bits 0 and 1 can be applied to the inputs of a gate by allowing a low voltage to represent logic 0 and a high voltage logic 1. This is called positive logic. In negative logic, the definitions are reversed.

The YES gate transmits the logic state (0 or 1) at its single input directly to its output. It's often used to interface logic circuits that are otherwise electronically incompatible. For this reason it's often called a *buffer*.

The NOT gate inverts or complements the logic state at its single input so it's often called an inverter. The NOT function is often indicated by a bar or vinculum over the symbol for an input or output that's been inverted. Thus if A is 0 and B is 1, then $A = \overline{B}$. (The $\overline{B}$ is read and sometimes written "not B.")

The AND gate is a decision making circuit with two or more inputs. The output of the AND gate is logic 0 unless *all* the inputs (inputs A *and* B *and* C . . .) are logic 1.

The OR gate is also a decision making circuit with two or more inputs. Its output is logic 0 unless *any* or *all* of its inputs (input A *or* B *or* C . . .) are 1.

The operation of a gate can be defined by a table that shows the combination of input bits that produces a particular output bit. Such a table is called a truth table. The truth tables and standard symbols for each of the four basic logic gates are shown in Fig. 1.

Compound Logic Circuits. Combining two or more of the basic gates into a compound logic circuit can provide some very important operating features. The two most important compound logic circuits are the AND-NOT and OR-NOT combinations. These are called the NAND and NOR gates and their symbols and truth tables are shown in Fig. 2.

As shown in Fig. 3, various combinations of NAND (or NOR) gates *alone* can simulate YES, NOT, and AND circuits. This is important, but the most fascinating characteristic of the NAND and NOR functions is their logic equivalence. Thanks to a rule known as DeMorgan's theorem, a positive logic NAND gate is equivalent to a negative logic NOR gate and vice versa.

You can prove this for yourself by writing the appropriate truth tables and finding that they are indeed identical. DeMorgan's theorem simplifies digital logic to the point where combinations of only NAND gates or NOR gates can implement any logic function. Figure 4, for example, shows how NAND gates alone can implement both the OR and NOR functions. Notice how NAND gates are used as inverters to change the inputs from positive to negative logic.

Complex Logic Systems. Simple and compound gates can be tied together to implement a countless variety of logic functions. Some of the resulting logic systems contain only a handful of gates; others may use dozens or even hundreds of gates. All of these complex logic systems can be divided into two broad categories: combinational and sequential.

Combinational circuits are characterized by their fast acting operation. Exclusive of the brief time delay required for its gates to react to an incoming logic 0 or 1 (the propagation time), the output(s) of the most complex combinational circuit instantaneously reflects the pattern of 0's and 1's at its input(s).

Sequential circuits include storage or delay elements that permit the logic result of a previous input to directly influence a new input. This makes sequential circuits slower than combinational circuits. But it also makes possible important applications such as memory registers, counters, dividers, sequencers, and microprocessors.

Combinational Logic Circuits. The simplest combinational logic circuit is the Exclusive-OR gate. The symbol and truth table for this circuit are shown in Fig. 5.

Look at the Exclusive-OR truth table for a moment. The Exclusive-OR function is just that; it gives a logic 1 output *only* if one *or* the other of its two inputs is logic 1. Otherwise the output is 0. This is identical to the binary addition rules with the exception of the carry output needed for 1 + 1.

It's easy to generate the carry output bit needed to use the Exclusive-OR circuit as a binary adder. Look at the logic circuit for an Exclusive-OR in Fig. 6. If you'll study the operation of this circuit, you'll find that the output of AND gate 1 provides the carry output we need. In the other circuit in Fig. 6, we use this carry output to form a circuit that can add any two binary bits. It's known as a *half adder*.

A half adder is useful, but it can only accept two input bits. To complete the binary addition rules, we need an adder circuit that will accept a carry bit as well. The circuit that accomplishes this goal is the *full adder*. As you can see in Fig. 7, a full adder can be made from two half adders and an OR gate.

It's possible to connect a string of adders together to form a binary adder capable of adding multiple-bit binary words. Figure 8, for instance, shows a 4-bit adder that will sum two words applied to its inputs. Try adding 1101 + 0101 using this adder to prove to yourself it really adds.

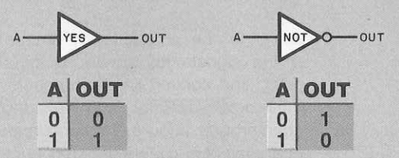
A binary adder forms part of a microprocessor's arithmetic-logic unit (ALU), a combinational circuit that performs addition, subtraction, and various logic operations upon two incoming words. The ALU is instructed what operations it is to perform by binary signals applied to its control inputs. We'll learn more about the ALU later in this course.

Encoders and Decoders. An encoder is a combinational network of OR gates that converts or encodes a nonbinary input into binary. For example, an octal-to-binary encoder has eight inputs (one for each octal digit) and three outputs (one for each binary bit). A logic 1 at one of the inputs produces the binary equivalent at the output.

Encoders can provide other conversion operations, too. Keyboard encoders, for instance, convert individual key positions into their assigned binary words. An example is the ASCII (*A*merican *S*tandard *C*ode for *I*nformation *I*nterchange) encoded keyboard, which generates the 7-bit word 0100101 when the % key is pressed.

A decoder is a combinational circuit that converts a binary number at its inputs into a logic 1 at one or more of its outputs. In digital electronics it's often necessary to convert a binary number into some other format, and one common decoder application is the conversion of binary numbers into the format required to activate the appropriate segments in a 7-segment decimal display.

Decoders are also used to decode binary instructions in a microprocessor,



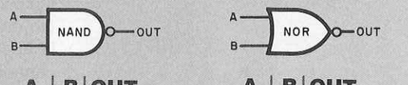
A	OUT
0	0
1	1

A	OUT
0	1
1	0

A	B	OUT
0	0	0
0	1	0
1	0	0
1	1	1

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 1. The four basic logic gates: YES, NOT, AND, OR.



A	B	OUT
0	0	1
0	1	1
1	0	1
1	1	0

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

Fig. 2. A NAND gate and a NOR gate, with their repective truth tables below.

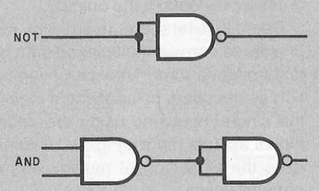


Fig. 3. Using NAND gates to simulate other gates. At top, YES; middle, NOT; bottom, AND.

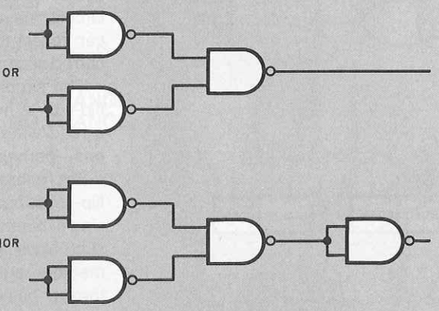
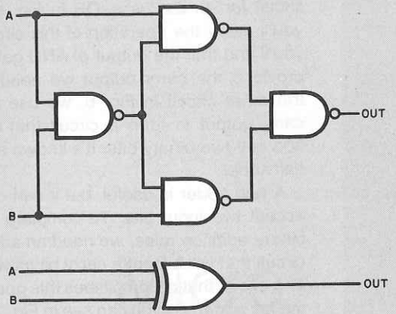


Fig. 4. Using NAND gates to prove DeMorgan's theorem. At top, OR; on the bottom, NOR.



A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 5. The combinational circuit at top provides an Exclusive-OR, as shown in the middle. The truth table is below.

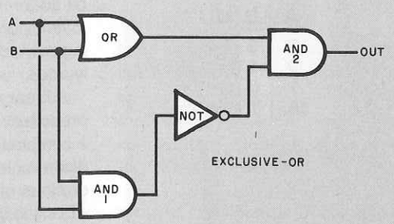
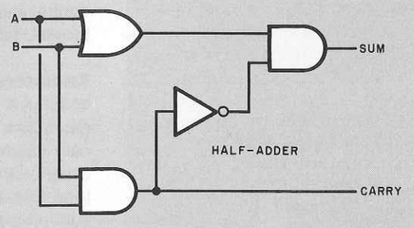


Fig. 6. How an Exclusive-OR can be used to make a half-adder combinational circuit.



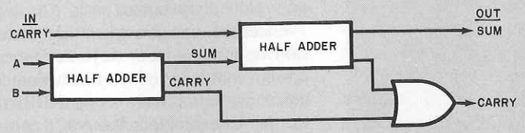


Fig. 7. Two half adders and an OR gate can be used to make a full adder circuit.

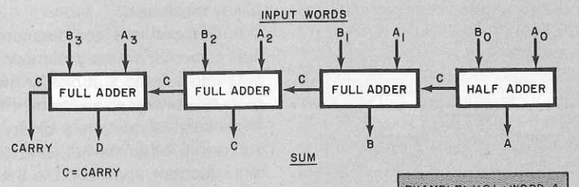


Fig. 8. A string of full adders connected together can be used to form a binary adder, capable of adding binary words.

assist in the production of sequential timing signals for advanced logic circuits, and convert binary numbers into their octal, decimal, and hexadecimal counterparts. Figure 9 summarizes the operation of encoders and decoders.

Multiplexers and Demultiplexers. The multiplexer is the digital logic equivalent of a multiple-position rotary switch. A typical multiplexer is a combinational logic circuit that selects one of several input lines and applies any data on that line to a single output. A special set of address inputs determines which input line is selected.

One typical multiplexer has eight data inputs, three address inputs, and a single data output. When the address 101 is applied to the multiplexer, input 5 is connected to the output.

A common application for multiplexers is driving the readouts of pocket calculators to reduce the number of pin connections on the calculator's chip. The multiplexer lets all the digits in the readout share a common set of terminals. It activates each digit or one segment in all the digits in rapid succession to fool the eye into thinking the display is continually illuminated.

The demultiplexer transfers the binary data at its input onto one of two or more output lines. Like the multiplexer, an address input controls the output.

Demultiplexers are used with multiplexers to convert multiplexed data back to its original form. They can even function as decoders by applying a logic 1 to the single input and using the address inputs as data inputs. Figure 10 summarizes the operation of multiplexers and demultiplexers.

Sequential Logic Circuits. Unlike combinational logic circuits, sequential circuits have memory. Their output(s) can reflect the effect of an input that occurred seconds or even days earlier.

The simplest sequential circuit is the flip-flop. A microprocessor together with a read/write memory incorporates dozens—perhaps thousands—of flip-flops.

There are several different kinds of flip-flops, but all are capable of storing a single binary bit. This makes possible such applications as counters, dividers, memory registers, and others. Here are the four basic kinds of flip-flops.

The RS Flip-Flop. The simplest flip-flop is made from two NAND or NOR gates with crisscrossed inputs and outputs as shown in Figure 11. This basic circuit is called a reset-set (RS) flip-flop

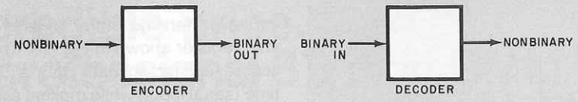


Fig. 9. An encoder is a combinational network that converts a nonbinary input to a binary output. A decoder does just the reverse.

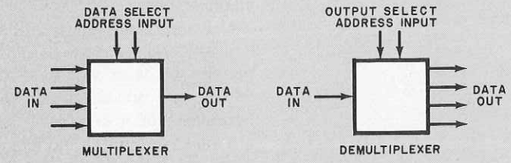
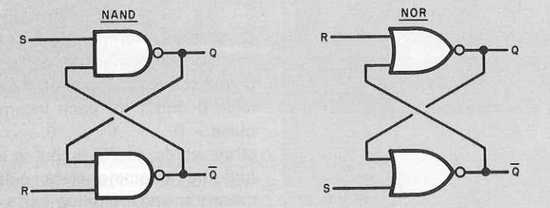


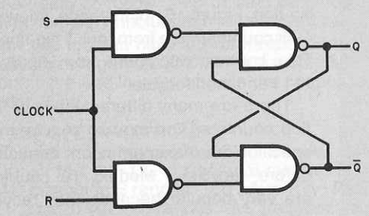
Fig. 10. A multiplexer is the equivalent of a multiple-position switch. A demultiplexer converts multiplexed data back to original form.



S	R	Q	Q̄
0	0	NOT ALLOWED	
0	1	1	0
1	0	0	1
1	1	NO CHANGE	

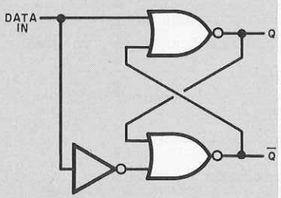
S	R	Q	Q̄
0	0	NO CHANGE	
0	1	0	1
1	0	1	0
1	1	NOT ALLOWED	

Fig. 11. Simplest flip-flop is made from two NAND's or two NOR's with truth tables as shown.



CLOCK	S	R	Q	Q̄
0	0	1	NO CHANGE	
0	1	0	NO CHANGE	
1	0	1	1	0
1	1	0	0	1

Fig. 12. A clocked RS flip-flop is a sequential circuit with truth table as shown here.



D	Q	Q̄
0	0	1
1	1	0

Fig. 13. A data, or D, flip-flop is made by adding an inverter to input of one flip-flop.

or simply a *latch*. Figure 11 also shows the truth tables for NAND and NOR gate versions of the RS flip-flop.

Notice that the two outputs of the RS flip-flop complement one another. When Q is logic 1, the flip-flop is *set*. When Q is logic 0, the flip-flop is *reset* or *cleared*.

Clocked RS Flip-Flop. The basic RS flip-flop is *asynchronous*; it responds to inputs as soon as they occur. A way to synchronize the operation of the RS flip-flop with other logic circuits is to gate its inputs so they can respond only when activated by a logic 1 from a *clock*. A clock is a sequential circuit that produces a stream of alternating 0's and 1's. Fig. 12 is a clocked RS flip-flop.

The Data or D Flip-Flop. The D flip-flop is a further modification of the clocked RS flip-flop. As shown in Figure 13, an inverter is added to one of the two inputs of the flip-flop and the remaining input and the inverter's input are tied together. This guarantees that the inputs to the RS section of the flip-flop will always complement one another. And it insures that the logic state of the Q output will always correspond to the logic state of the D input.

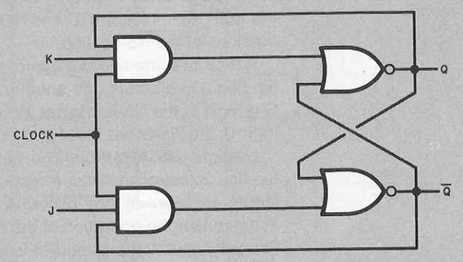
The JK Flip-Flop. The JK flip-flop is a clocked RS flip-flop with a refinement that allows a logic 1 to be simultaneously applied to both inputs. Figure 14 shows the logic circuit and truth table for this flip-flop. The JK flip-flop can easily simulate any of the other kinds of flip-flops, so it's commonly used in sequential logic circuits.

The JK flip-flop can be used to make a toggle or T flip-flop. The J and K inputs are tied together and called the T input. When a logic 1 is applied to T, the flip-flop changes state or toggles each time a clock pulse arrives.

Storage Registers. A string of D flip-flops called a *register* can be used to store a binary word. A register like this can be made far more useful by adding some combinational logic to simultaneously clear all the flip-flops to 0 when a logic 1 is applied to a *clear* input. A *load* input can also be added to force the register to ignore incoming data. When the load input is logic 1, the input data will be accepted by the register when the next clock pulse arrives.

Data storage registers like this are sometimes called *buffer* registers. They're used in logic circuits and in microprocessor units to temporarily hold a data word.

Shift Registers. Considerably more



Q=0

J	K	Q*
0	0	0
0	1	0
1	0	1
1	1	1

*(after clock pulse)

Q=1

J	K	Q*
0	0	1
0	1	0
1	0	1
1	1	0

*(after clock pulse)

Fig. 14. The JK flip-flop is a clocked RS flip-flop that allows a logic 1 to be simultaneously applied to both inputs. Shown here is a NOR gate version with truth tables for Q = 0 and Q = 1.

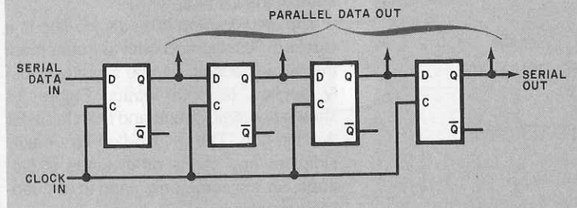


Fig. 15. This shift register made from D flip-flops accepts data a bit at a time and has a serial output as well as parallel outputs from each flip-flop.

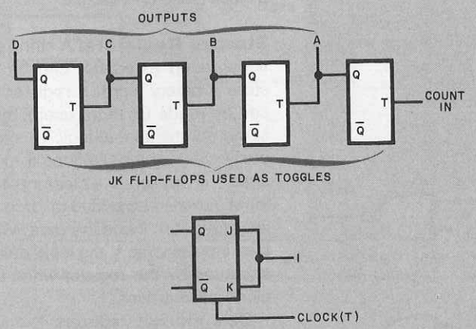


Fig. 16. A four-bit counter made from T flip-flops that will count from 0000 to 1111 and then recycle.

versatile than the buffer register is the shift register shown in Fig. 15. This particular register accepts data a bit at a time (serial input) while making available the contents of all its flip-flops simultaneously (parallel output). The data bits in the register are shifted right a bit at a time by clock pulses to make room for incoming bits.

Universal shift registers that can accept and output data as serial bits or parallel words as well as shift the data left or right are available. The various operations of a universal shift register are selected by applying logical 0's and 1's to an array of control inputs. Microprocessors incorporate at least one shift register to perform some of the data manipulation required to multiply and divide binary numbers.

Counters. Remember the toggle or T flip-flop we discussed earlier? The Q output of this flip-flop alternates between logic 0 and 1 for each incoming clock pulse: 0 . . . 1 . . . 0. . . 1 . . . In other words, the Q output is logic 1 for *half* the incoming clock pulses. This means a single flip-flop can be used to divide an incoming stream of bits by two. The Q output of a toggle flip-flop also counts! Thus, 0 . . . 1 . . . 0 . . . 1 . . . is the same as counting from 0 to 1 in binary over and over again.

Higher capacity binary counters (and dividers) can be made from a string of T flip-flops. Just connect the Q output of one flip-flop to the clock input of the next flip-flop. Figure 16, for instance, shows a 4-bit counter made from four T flip-flops. This counter will count from 0000 to 1111 and then recycle.

There are many different kinds of flip-flop counters. The *modulo* of a counter specifies the maximum count it reaches before recycling. Modulo 10 counters are very popular because they recycle after the tenth input pulse and therefore provide a convenient way to count in decimal. They are often called BCD (*bi*nary *c*oded *d*ecimal) counters. Their count sequence is 0000 (0_{10}) . . . 0001 (1_{10}) . . . 0010 (2_{10}) . . . . . 1001 (9_{10}) . . . 0000 (0_{10}) . . . .

Counters can have a variety of control inputs. A typical counter, for example, can count up *or* down. It may also have control inputs for clearing the count to all 0's, presetting the count to any desired value, and enabling the counter to count. Finally, since counters store the accumulated count until the next clock pulse arrives, they can be considered storage registers. ◇