New Logic Function Symbols

Here is what these latest graphic symbols mean and why they are now being used in many data books and schematics

By Jan Axelson

To the surprise (and confusion) of many readers who refer to manufacturers' data books for information, different graphic symbols are used to portray logic functions. In these new symbols the familiar shapes of AND and OR gates may not be seen. Instead, digital logic can be illustrated by rectangular symbols that contain a variety of other special symbols and notations. Texas Instruments, for example, use these modern symbols in their data books while also illustrating logic diagrams with the well-known symbology.

What are the new graphic symbols, and why and where are they used? The following article answers these questions. In addition, by decoding the symbols for three different ICs, you'll learn how to go about interpreting the graphics symbol for just about any logic chip.

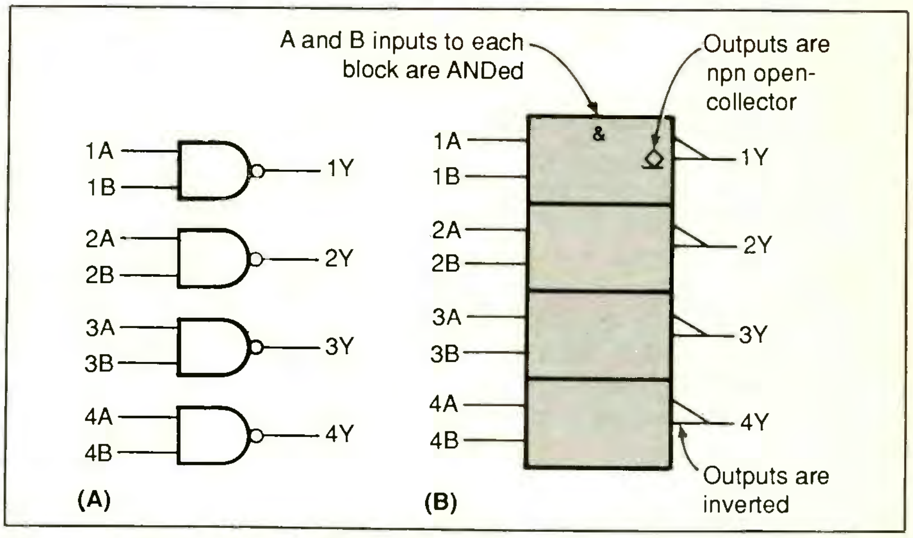

A and B inputs to each block are ANDed
Outputs are npn open-collector
1A
1B
1Y
2A
2B
2Y
3A
3B
3Y
4A
4B
4Y
&
Outputs are inverted
(A)
(B)

Fig. 1. The newer logic symbol in (B) represents the same device as the logic diagram in (A) does. Each rectangle in the logic symbol represents a dual-input NAND gate.

A New Logic Language

Why develop a whole new graphic language when one already exists? The reason rests on the fact that functions and capabilities of integrated circuits are becoming more and more complex. Consequently, it's becoming very difficult to draw a logic diagram that describes both fully and completely how a particular IC behaves.

Realizing this, the IEEE (Institute of Electrical and Electronics Engineers) developed a new standard for symbols that could show the functions and behavior of any logic circuit much more concisely. The new standard (ANSI/IEEE Std 91-1984 and IEC Publication 617-12) was written to be compatible with the recommendations of the IEC (International Electrotechnical Commission), whose standards are followed world-wide. Contributors included members of industry, government, and education.

The new logic symbols can portray the functions of complicated logic circuitry in detail, yet compactly. In contrast, "old-style" logic diagrams often require a maze of connections among logic gates and other circuit elements. To understand how a device operates, you often find yourself laboriously tracing a signal through the maze.

A space-saving alternative way to portray a device is to draw a simple rectangle and label the inputs and outputs. But this doesn't tell you very much about how the circuit behaves—you have to go back to the data sheet to find out more.

The new logic symbols offer a third choice. They're concise because many functions can be described by a single symbol within the main logic symbol, and related signals are coded to show how they affect each other. Also, logic symbols can give you more information—there are notations to represent open-collector, open-emitter, and three-state outputs, for instance.

(From here on, I'll refer to the "old-style" drawings as logic diagrams, and to the "new-style" drawings as logic symbols.)

The official document describing the new symbols is called Graphic Symbols for Logic Functions. As standards go, this one is easy to understand and use. It includes full definitions of terms used, many examples, and much more detail than this brief article does.

Single copies of the full standard cost $12 plus $4 shipping and handling, from the American National Standards Institute, Attention: Sales Department, 1430 Broadway, New York, NY 10018. Ask for ANSI/IEEE standard 91-1984. Also, a helpful summary of the symbols and their meanings can be found in *The TTL Data Book*, Volume 1, 1984, from Texas Instruments.

Decoding a Symbol

How do you interpret the new symbols? A logic symbol is made up of two, or sometimes three, elements: the outline, qualifying symbols and, optionally, dependency notation. Each of these elements has its own symbolic language. In addition, inputs and outputs are normally labeled on the symbol.

Let's look at an example. Figure 1 shows both the logic diagram and the logic symbol for a 7403 quadruple NAND gate. The logic diagram consists of four familiar NAND gate symbols and their inputs and outputs.

In the logic symbol, the outline is four rectangles stacked vertically. Unless arrows indicate otherwise, inputs are assumed to be, as here, on the left, and outputs on the right. The particular dimensions of the rectangles aren't significant. However, the fact that the rectangles are stacked vertically—perpendicular to the left-to-right signal flow—*is* significant. It tells you that there is no logic connection between the blocks and that each block functions independently.

The logic symbol for the 7403 uses three different qualifying symbols. Qualifying symbols are notations that describe logic characteristics affecting the inputs, outputs or overall characteristics of a logic circuit.

Table I shows some of these as specified by the standard. Many of the symbols represent functions that

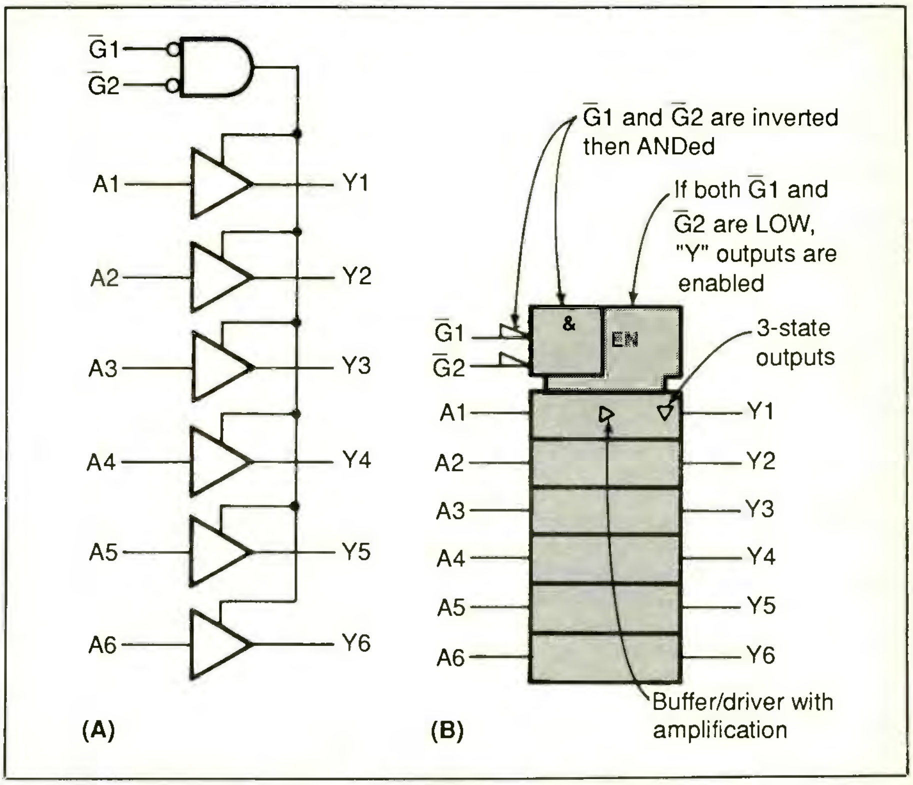

$\overline{G}$1
$\overline{G}$2
A1
A2
A3
A4
A5
A6
Y1
Y2
Y3
Y4
Y5
Y6
(A)
$\overline{G}$1 and $\overline{G}$2 are inverted then ANDed
If both $\overline{G}$1 and $\overline{G}$2 are LOW, "Y" outputs are enabled
&
EN
3-state outputs
Buffer/driver with amplification
(B)

Fig. 2. The familiar logic diagram for a 74365 hex bus driver is shown in (A), the logic symbol in (B). The distinctively shaped common control block in the logic symbol contains the signals that enable the six drivers.

are difficult to portray concisely using the traditional logic gates. By using a qualifying symbol, you can describe a logic operation without showing every circuit element and connection involved.

One qualifying symbol in Fig. 1 is easy to guess the meaning of: the & means that the two inputs to each block are ANDed. If there is little chance for confusion, a symbol that applies to all blocks in a row need be included only in the top block. In this example, you can assume that the & applies to all four rectangles.

The triangular symbols at each output tell you the outputs are active-LOW, or inverted. Combine these inverters with the AND function and you have four NAND gates.

The other qualifying symbol in the top block indicates NPN open-collector outputs. This is the kind of information that is helpful to have included in the logic symbol; you no longer have to search through the data sheet to ferret out such details.

From this example, you can come up with ways to portray similar circuits. To draw the symbol for a quad NOR gate, replace the AND symbol with the symbol for an OR gate. To show AND gates, leave out the inverting triangles. And so on.

Pin numbers can be added to the symbol if desired. Power supply connections can also be shown, using the symbol for non-logic connections (see Table I).

Finally, you may be relieved to learn that the NAND-gate symbols in 1A aren't forbidden by the new standard. Either system is allowed—it's up to you to choose the appropriate symbols for a particular situation.

Common Control Blocks

Drawing or reading the logic sym-

Table I—General Symbols

GENERAL SYMBOLS

SYMBOL	FUNCTION
&	AND gate
≥ 1	OR gate
= 1	EXCLUSIVE-OR gate
▷	Buffer/driver with amplification
⎎	Schmitt trigger; input with hysteresis
X/Y	Coder (example: BIN/BCD)
MUX	Multiplexer
DMUX	Demultiplexer
Σ	Adder
⎍	Retriggerable monostable
G ⎍⎍	Astable element
SRGm	Shift register; m = number of bits (example: SRG 4)
CTRm	Counter; m = number of bits (example: CTR 8)
ROM*	Read-only memory; * = number of addresses and bits (example: ROM 256 X 4)
RAM*	Random-access memory, *number of addresses and bits (example: RAM 256 X 1)
—○	Negated input (external 0 causes internal 1) specify (positive logic: LOW = 0, HIGH = 1) or (negative logic: HIGH = 0, LOW = 1)
—◺	Active-LOW input (same as —○ in positive logic)
—▷	Dynamic input, transition causes temporary logic 1
J	J input to JK flip-flop
K	K input to JK flip-flop
R	Reset
S	Set
T	Toggle
D	Data input to storage element (flip-flop, shift register, etc.)
EN	Enable input
—⊓	Analog input
—✕	Non-logic connection (such as to power supplies)

SYMBOLS DESCRIBING OUTPUTS

SYMBOL	FUNCTION
○—	Negated output (internal 1 causes external 0) (specify positive logic or negative logic)
◺—	Active-LOW output (same as ○— in positive logic)
◇	NPN open-collector output, or equivalent
◇	NPN open-emitter output, or equivalent
▽	Three-state output
¬	Postponed output (change in state delayed until initializing input returns to its original state)

bols for simple logic gates is relatively straightforward. A symbol that's a little more complex is the one describing a 74365 hex bus driver, shown along with its logic diagram in Fig. 2. The outline of the logic symbol is made up of six matching rectangles topped off by a distinctively shaped common control block. And the common control block has another rectangle embedded in it.

The common control block is identifiable by its indentations at the bottom (or top, if that's where it attaches to the rest of the circuit). The inputs to a common control block affect the signals in the blocks in the row attached to the control block's indented end.

The symbol shows that G1 and G2, the inputs to the control block, are inverted, and the & tells you that the two inverted inputs are then ANDed inside the chip.

Note that the two blocks making up the common control block have a vertical side in common. This tells you that there *is* a logic connection between the blocks. When the output of the AND function is active (logic 1), the Enable function is also active, and the six Y outputs are enabled.

Common control blocks might have any of a number of functions besides Enable. Counter (CTR), Shift Register (SRG), and Multiplexer (MUX), all functions that may affect several circuit elements at once, are just a few.

Two other qualifying symbols complete the logic symbol. The triangular symbol in the center signifies a buffer or driver with amplification. The downward-pointing triangular symbol at Y1 tells you that the outputs are three-state: When enabled, they are either HIGH or LOW. Otherwise, they are in a state of high impedance. Again, the symbols in the top driver block apply to the blocks below them—each of the six blocks represents a driver with three-state output.

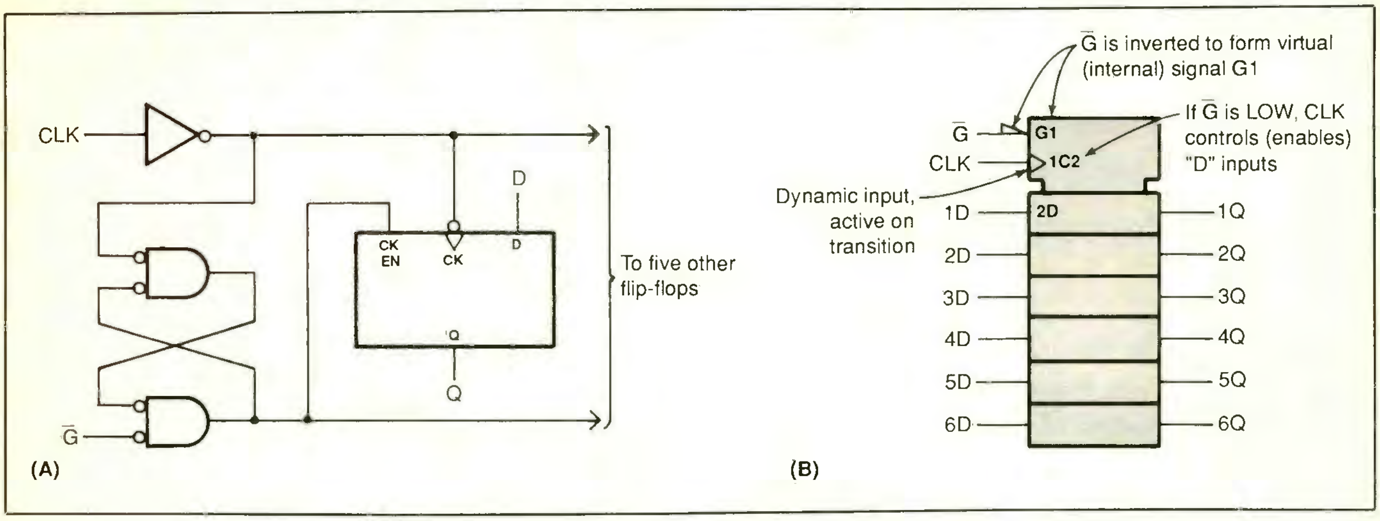
Dependency Notation

Figure 3 shows the logic diagram and logic symbol for a 74LS378 hex D-type flip-flop. This symbol introduces the one type of notation we haven't yet seen—dependency notation. Dependency notation describes functions that affect only some of a circuit's signals.

Many functions, such as AND and Enable, can be portrayed with either qualifying symbols or dependency notation. But dependency notation is unique because each notation includes a code to indicate which inputs and outputs are affected. If an enable input affects a particular signal, the enabling input can be labeled EN1 and the enabled signal labeled 1 to show the relationship.

The position of the numeric code in the label distinguishes the affecting, or controlling, signal from its affected, or controlled, signals. In the affecting signal, the identifying number follows the dependency label (as in C1). In the affected signal, it precedes the signal label, if one is present (as in 1D). Any convenient numeric code can be used, as long as a different number is chosen for each dependency effect in the symbol.

In all, eleven kinds of dependency



CLK
D
CK EN
CK
D
Q
Q
To five other flip-flops
G
(A)
G is inverted to form virtual (internal) signal G1
If G is LOW, CLK controls (enables) "D" inputs
G
CLK
G1
1C2
Dynamic input, active on transition
1D
2D
2D
3D
4D
5D
6D
1Q
2Q
3Q
4Q
5Q
6Q
(B)

Fig. 3. In the logic symbol for the 74LS378 hex D-type flip-flop, signals G1, 1C2, and D2 are coded to show which signals they affect and/or are affected by.

notation are specified by the standard. They're listed in Table II. The symbol for the 74LS378 uses two of these: AND (G) and Control (C) Dependency.

Examining the symbol tells you just what effects the dependency notations describe in the circuit. The label G1 describes the virtual, or internal, signal created by the negated G input. Inside the outline, signals labeled 1 are active only when G is LOW. The affected signal in this case is 1C2, at the CLK input.

Table II—Dependency Notations

Symbol	Type	Result when affecting input = Logic 1	Logic 0	Comments
Boolean Functions				
G	AND	Does not alter state	Logic 0	Acts like an AND gate
V	OR	Logic 1	Does not alter state	Acts like an OR gate
N	Negate	Complements logic state	Does not alter state	Acts like an Exclusive-OR gate
Enable Functions				
C	Control	Permits action	Prevents action	Produces an action
EN	Enable	Permits action	Prevents action	Has a single preparatory effect
M	Mode	Permits action	Prevents action	Has alternative effects
A	Address	Permits action	Prevents action	Selects an address
Bistable Functions				
R	Reset	Acts as if S = 0, R = 1	No effect	In bistables, specifies result when R = S = 1
S	Set	Acts as if S = 1, R = 0	No effect	
Other Functions				
X	Transmit	Path established	No path established	Forms transmission path
Z	Interconnect	Logic 1	Logic 0	Indicates a logic connection

In turn, the 2 in 1C2 tells you that CLK controls inputs labeled 2D inside the outline. Again, because the bottom five blocks are unlabeled, you can assume that the notation 2D applies to all six flip-flops. (Be careful here not to confuse the qualifying symbol 2D, which applies to each of the six flip-flops, with the input label 2D, which identifies the second flip-flop.)

The triangular symbol at CLK means that CLK is a dynamic input: When CLK changes state from LOW to HIGH, the rising edge of CLK causes a transitory, or temporary, logic 1 at 1C2. So in all, the symbol tells you that when G is low, the rising edge of CLK transfers the data at the six inputs to their corresponding outputs.

Other types of dependency notation show OR, exclusive-OR, mode select (such as count up and count down), and other functions. The R and S dependency notations specify what happens in a bistable when both Set and Reset are HIGH. An output that is affected by two or more dependencies is analyzed by determining the effects of the dependencies from left to right as shown on the symbol.

(Continued on page 92)

In short, any time a function affects only some of the signals in a logic function, dependency notation can concisely specify which signals are affected.

Decoding Other Symbols

These examples illustrate how to interpret the new logic symbols. In practice such relatively simple and familiar functions really don't require the new symbols to portray them clearly. In general, the more complicated the device, the more likely it is that a logic symbol will help.

Any logic symbol can be decoded using the same techniques as were used with these three ICs. First, examine the blocks that make up the symbol and look for a common control block. Then determine how the qualifying and dependency symbols affect the inputs and outputs. Many, and hopefully most, of the symbols used in the standard are already familiar or are easy to guess the meanings of. Once you've mastered the system, the specifics aren't difficult to pick up.

Learning a new language always involves some effort in learning the rules and a new vocabulary, but your reward is being able to communicate in new and often improved ways. This new language is no exception.