To permit a simplified "worst-case" analysis, designers often use a single *worst-case delay* specification that is the maximum of t_{pLH} and t_{pHL} specifications. The worst-case delay through a circuit is then computed as the sum of the worst-case delays through the individual components, independent of the transition direction and other circuit conditions. This may give an overly pessimistic view of the overall circuit delay, but it saves design time and it's guaranteed to work.

worst-case delay

5.2.5 Timing Analysis Tools

Sophisticated CAD tools for logic design make timing analysis even easier. Their component libraries typically contain not only the logic symbols and functional models for various logic elements, but also their timing models. A simulator allows you to apply input sequences and observe how and when outputs are produced in response. You typically can control whether minimum, typical, maximum, or some combination of delay values are used.

Even with a simulator, you're not completely off the hook. It's usually up the designer to supply the input sequences for which the simulator should produce outputs. Thus, you'll need to have a good feel for what to look for and how to stimulate your circuit to produce and observe the worst-case delays.

Some timing analysis programs can automatically find all possible delay paths in a circuit, and print out a sorted list of them, starting with the slowest. These results may be overly pessimistic, however, as some paths may actually not be used in normal operations of the circuit; the designer must still use some intelligence to interpret the results properly.

5.3 Combinational PLDs

5.3.1 Programmable Logic Arrays

Historically, the first PLDs were *programmable logic arrays (PLAs).* A PLA is a combinational, two-level AND-OR device that can be programmed to realize any sum-of-products logic expression, subject to the size limitations of the device. Limitations are

programmable logic array (PLA)

- the number of inputs (n), *inputs*
- the number of outputs (m), and *outputs*
- the number of product terms (p). *product terms*

We might describe such a device as "an $n \times m$ PLA with p product terms." In general, p is far less than the number of n-variable minterms (2^n). Thus, a PLA cannot perform arbitrary n-input, m-output logic functions; its usefulness is limited to functions that can be expressed in sum-of-products form using p or fewer product terms.

An $n \times m$ PLA with p product terms contains p $2n$-input AND gates and m p-input OR gates. Figure 5-21 shows a small PLA with four inputs, six AND

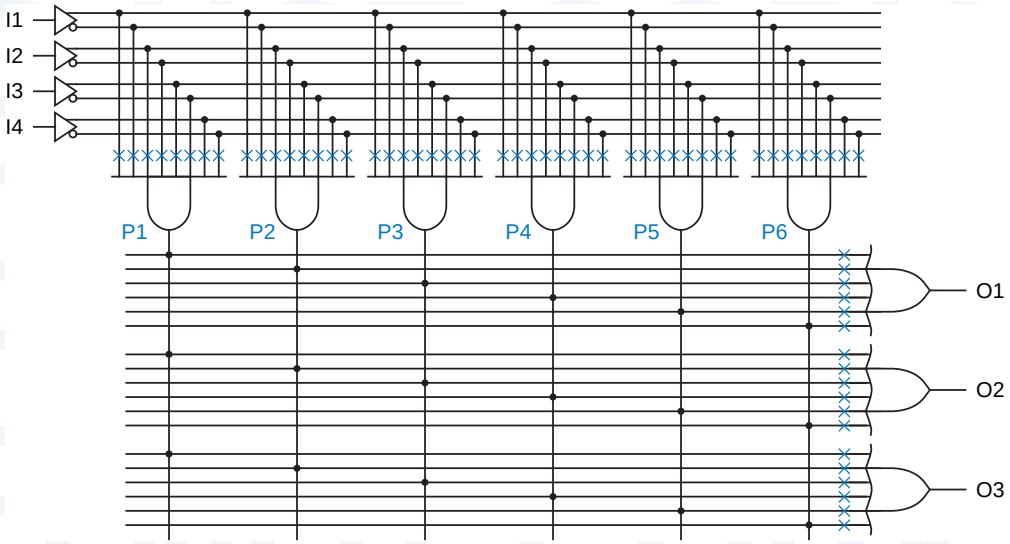


Figure 5-21 A 4 × 3 PLA with six product terms.

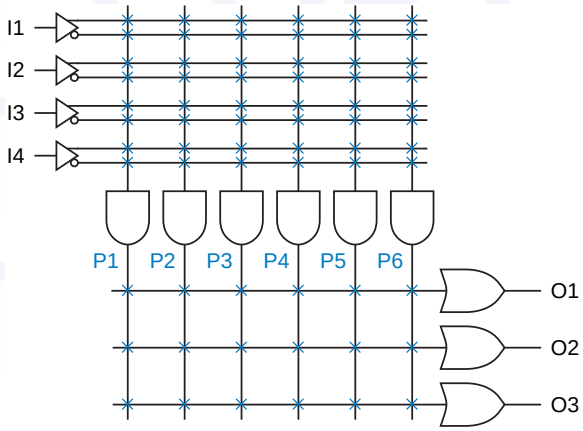
PLD fuses

gates, and three OR gates and outputs. Each input is connected to a buffer that produces both a true and a complemented version of the signal for use within the array. Potential connections in the array are indicated by X's; the device is programmed by establishing only the connections that are actually needed. The needed connections are made by *fuses*, which are actual fusible links or non-volatile memory cells, depending on technology as we explain in Sections 5.3.4 and 5.3.5. Thus, each AND gate's inputs can be any subset of the primary input signals and their complements. Similarly, each OR gate's inputs can be any subset of the AND-gate outputs.

PLA diagram

As shown in Figure 5-22, a more compact diagram can be used to represent a PLA. Moreover, the layout of this diagram more closely resembles the actual internal layout of a PLA chip (e.g., Figure 5-28 on page 308).

Figure 5-22
Compact representation of a 4 × 3 PLA with six product terms.



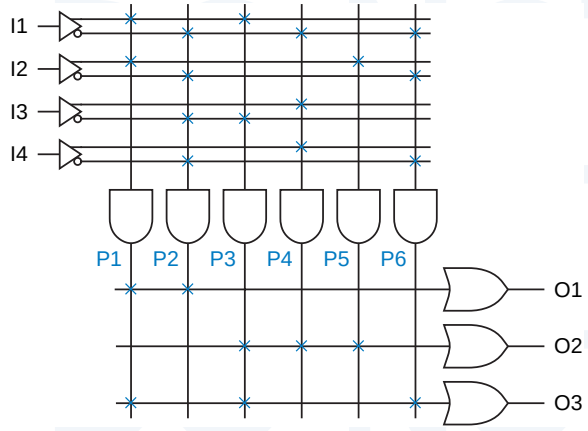


Figure 5-23
A 4 × 3 PLA programmed with a set of three logic equations.

The PLA in Figure 5-22 can perform any three 4-input combinational logic functions that can be written as sums of products using a total of six or fewer distinct product terms, for example:

$$O1 = I1 \cdot I2 + I1' \cdot I2' \cdot I3' \cdot I4'$$
$$O2 = I1 \cdot I3' + I1' \cdot I3 \cdot I4 + I2$$
$$O3 = I1 \cdot I2 + I1 \cdot I3' + I1' \cdot I2' \cdot I4'$$

These equations have a total of eight product terms, but the first two terms in the O3 equation are the same as the first terms in the O1 and O2 equations. The programmed connection pattern in Figure 5-23 matches these logic equations.

PLA constant outputs

Sometimes a PLA output must be programmed to be a constant 1 or a constant 0. That's no problem, as shown in Figure 5-24. Product term P1 is always 1 because its product line is connected to no inputs and is therefore always pulled HIGH; this constant-1 term drives the O1 output. No product term drives the O2 output, which is therefore always 0. Another method of obtaining a constant-0 output is shown for O3. Product term P2 is connected to each input variable and its complement; therefore, it's always 0 ($X \cdot X' = 0$).

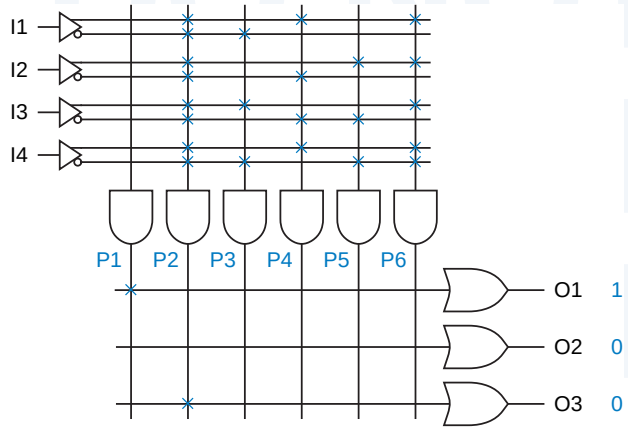


Figure 5-24
A 4 × 3 PLA programmed to produce constant 0 and 1 outputs.

AN UNLIKELY GLITCH

Theoretically, if *all* of the input variables in Figure 5-24 change *simultaneously*, the output of product term P2 could have a brief 0-1-0 glitch. This is highly unlikely in typical applications, and is impossible if one input happens to be unused and is connected to a constant logic signal.

Our example PLA has too few inputs, outputs, and AND gates (product terms) to be very useful. An n-input PLA could conceivably use as many as 2^n product terms, to realize all possible n-variable minterms. The actual number of product terms in typical commercial PLAs is far fewer, on the order of 4 to 16 per output, regardless of the value of n.

The Signetics 82S100 was a typical example of the PLAs that were introduced in the mid-1970s. It had 16 inputs, 48 AND gates, and 8 outputs. Thus, it had $2 \times 16 \times 48 = 1536$ fuses in the AND array and $8 \times 48 = 384$ in the OR array. Off-the-shelf PLAs like the 82S100 have since been supplanted by PALs, CPLDs, and FPGAs, but custom PLAs are often synthesized to perform complex combinational logic within a larger ASIC.

5.3.2 Programmable Array Logic Devices

programmable array logic (PAL) device

A special case of a PLA, and today's most commonly used type of PLD, is the *programmable array logic (PAL) device*. Unlike a PLA, in which both the AND and OR arrays are programmable, a PAL device has a *fixed* OR array.

The first PAL devices used TTL-compatible bipolar technology and were introduced in the late 1970s. Key innovations in the first PAL devices, besides the introduction of a catchy acronym, were the use of a fixed OR array and bidirectional input/output pins.

PAL16L8

These ideas are well illustrated by the *PAL16L8*, shown in Figures 5-25 and 5-26 and one of today's most commonly used combinational PLD structures. Its programmable AND array has 64 rows and 32 columns, identified for programming purposes by the small numbers in the figure, and $64 \times 32 = 2048$ fuses. Each of the 64 AND gates in the array has 32 inputs, accommodating 16 variables and their complements; hence, the "16" in "PAL16L8".

FRIENDS AND FOES

PAL is a registered trademark of Advanced Micro Devices, Inc. Like other trademarks, it should be used only as an adjective. Use it as a noun or without a trademark notice at your own peril (as I learned in a letter from AMD's lawyers in February 1989).

To get around AMD's trademark, I suggest that you use a descriptive name that is more indicative of the device's internal structure: a *fixed-OR element (FOE).*

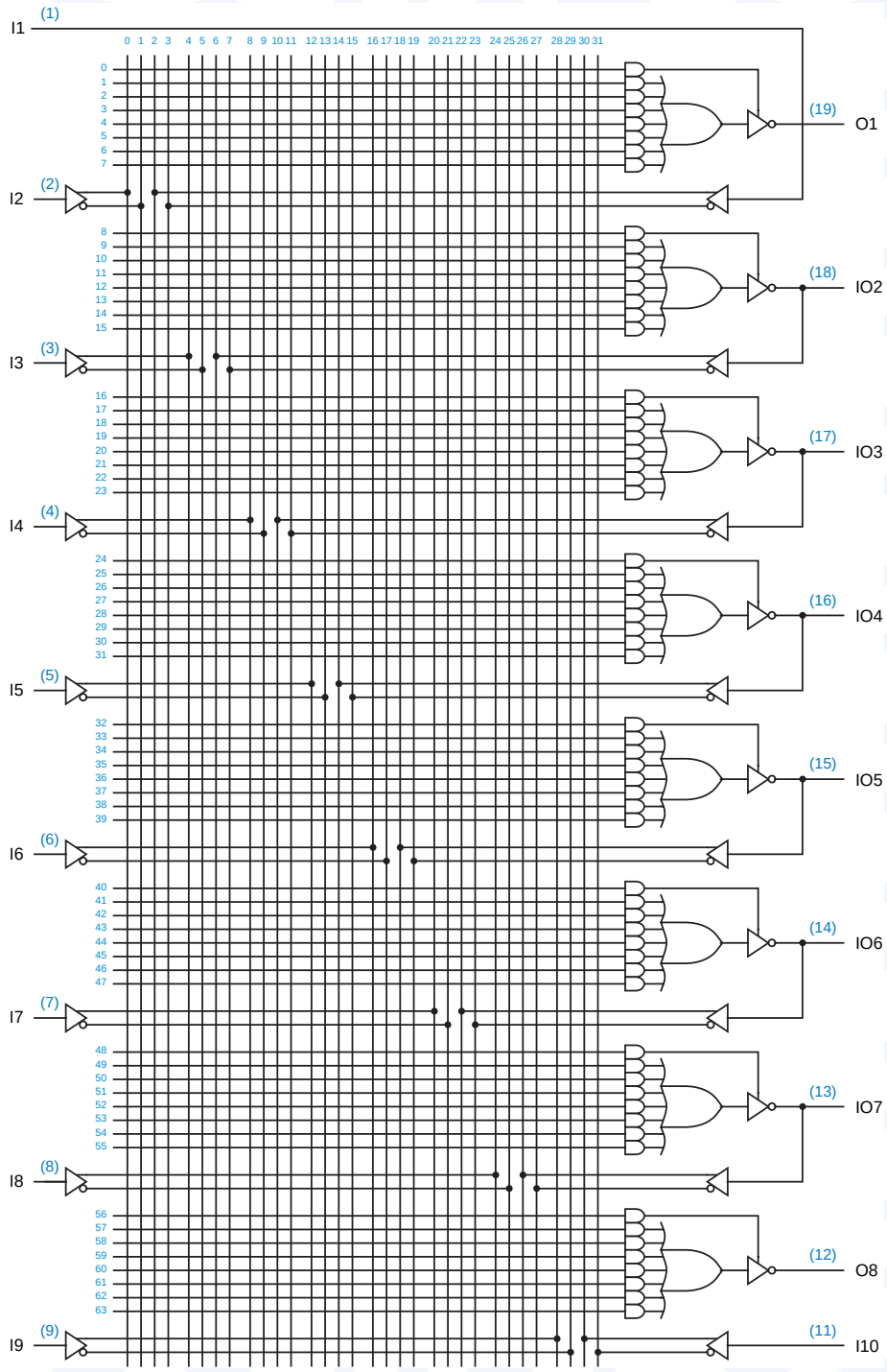


Figure 5-25 Logic diagram of the PAL16L8.

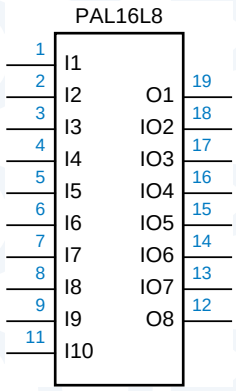


Figure 5-26
Traditional logic symbol for the PAL16L8.

output-enable gate

Eight AND gates are associated with each output pin of the PAL16L8. Seven of them provide inputs to a fixed 7-input OR gate. The eighth, which we call the *output-enable gate*, is connected to the three-state enable input of the output buffer; the buffer is enabled only when the output-enable gate has a 1 output. Thus, an output of the PAL16L8 can perform only logic functions that can be written as sums of seven or fewer product terms. Each product term can be a function of any or all 16 inputs, but only seven such product terms are available.

Although the PAL16L8 has up to 16 inputs and up to 8 outputs, it is housed in a dual in-line package with only 20 pins, including two for power and ground (the corner pins, 10 and 20). This magic is the result of six bidirectional pins (13–18) that may be used as inputs or outputs or both. This and other differences between the PAL16L8 and a PLA structure are summarized below:

- The PAL16L8 has a fixed OR array, with seven AND gates permanently connected to each OR gate. AND-gate outputs cannot be shared; if a product term is needed by two OR gates, it must be generated twice.
- Each output of the PAL16L8 has an individual three-state output enable signal, controlled by a dedicated AND gate (the output-enable gate). Thus, outputs may be programmed as always enabled, always disabled, or enabled by a product term involving the device inputs.

HOW USEFUL ARE SEVEN PRODUCT TERMS?

The worst-case logic function for two-level AND-OR design is an n-input XOR (parity) function, which requires 2^{n-1} product terms. However, less perverse functions with more than seven product terms of a PAL16L8 can often be built by decomposing them into a 4-level structure (AND-OR-AND-OR) that can be realized with two passes through the AND-OR array. Unfortunately, besides using up PLD outputs for the first-pass terms, this doubles the delay, since a first-pass input must pass through the PLD twice to propagate to the output.

COMBINATIONAL, NOT COMBINATORIAL!

A step *backwards* in MMI's introduction of PAL devices was their popularization of the word "combinatorial" to describe combinational circuits. *Combinational* circuits have no memory—their output at any time depends on the current input *combination*. For well-rounded computer engineers, the word "combinatorial" should conjure up vivid images of binomial coefficients, problem-solving complexity, and computer-science-great Donald Knuth.

- There is an inverter between the output of each OR gate and the external pin of the device.
- Six of the output pins, called *I/O pins*, may also be used as inputs. This provides many possibilities for using each I/O pin, depending on how the device is programmed: *I/O pin*
  - If an I/O pin's output-control gate produces a constant 0, then the output is always disabled and the pin is used strictly as an input.
  - If the input signal on an I/O pin is not used by any gates in the AND array, then the pin may be used strictly as an output. Depending on the programming of the output-enable gate, the output may always be enabled, or it may be enabled only for certain input conditions.
  - If an I/O pin's output-control gate produces a constant 1, then the output is always enabled, but the pin may still be used as an input too. In this way, outputs can be used to generate first-pass "helper terms" for logic functions that cannot be performed in a single pass with the limited number of AND terms available for a single output. We'll show an example of this case on page 325.
  - In another case with an I/O pin always output-enabled, the output may be used as an input to AND gates that affect the very same output. That is, we can embed a feedback sequential circuit in a PAL16L8. We'll discuss this case in \secref{palatch}.

The *PAL20L8* is another combinational PLD similar to the PAL16L8, except that its package has four more input-only pins and each of its AND gates has eight more inputs to accommodate them. Its output structure is the same as the PAL16L8's. *PAL16L8*

5.3.3 Generic Array Logic Devices

In \chapref{SeqPLDs} we'll introduce sequential PLDs, programmable logic devices that provide flip-flops at some or all OR-gate outputs. These devices can be programmed to perform a variety of useful sequential-circuit functions.

COMBINATIONAL PLD SPEED

The speed of combinational PLDs is usually stated as a single number that gives the propagation delay t_{PD} from any input to any output for either direction of transition. PLDs are available in a variety of speed grades; commonly used parts run at 10 ns. In 1998, the fastest available combinational PLDs included a bipolar PAL16L8 at 5 ns and a 3.3-V CMOS GAL22LV10 at 3.5 ns.

generic array logic
GAL device
GAL16V8

One type of sequential PLD, first introduced by Lattice Semiconductor, is called *generic array logic* or a *GAL device*, and is particularly popular. A single GAL device type, the *GAL16V8*, can be configured (via programming and a corresponding fuse pattern) to emulate the AND-OR, flip-flop, and output structure of any of a variety of combinational and sequential PAL devices, including the PAL16L8 introduced previously. What's more, the GAL device can be erased electrically and reprogrammed.

Figure 5-27 shows the logic diagram for a GAL16V8 when it has been configured as a strictly combinational device similar to the PAL16L8. This configuration is achieved by programming two "architecture-control" fuses, not shown. In this configuration, the device is called a *GAL16V8C*.

GAL16V8C

The most important thing to note about the GAL16V8C logic diagram, compared to that of a PAL16L8 on page 303, is that an XOR gate has been inserted between each OR output and the three-state output driver. One input of the XOR gate is "pulled up" to a logic 1 value but connected to ground (0) via a fuse. If this fuse is intact, the XOR gate simply passes the OR-gate's output unchanged, but if the fuse is blown the XOR gate inverts the OR-gate's output. This fuse is said to control the *output polarity* of the corresponding output pin.

output polarity

Output-polarity control is a very important feature of modern PLDs, including the GAL16V8. As we discussed in Section 4.6.2, given a logic function to minimize, an ABEL compiler finds minimal sum-of-products expressions for both the function and its complement. If the complement yields fewer product terms, it can be used if the GAL16V8's output polarity fuse is set to invert. Unless overridden, the compiler automatically makes the best selection and sets up the fuse patterns appropriately.

PALCE16V8
GAL20V8
PALCE20V8

Several companies make a part that is equivalent to the GAL16V8, called the *PALCE16V8*. There is also a 24-pin GAL device, the *GAL20V8* or *PALCE20V8*, that can be configured to emulate the structure of the PAL20L8 or any of a variety of sequential PLDs, as described in \secref{seqGAL}.

LEGAL NOTICE GAL is a trademark of Lattice Semiconductor, Hillsboro, OR 97124.

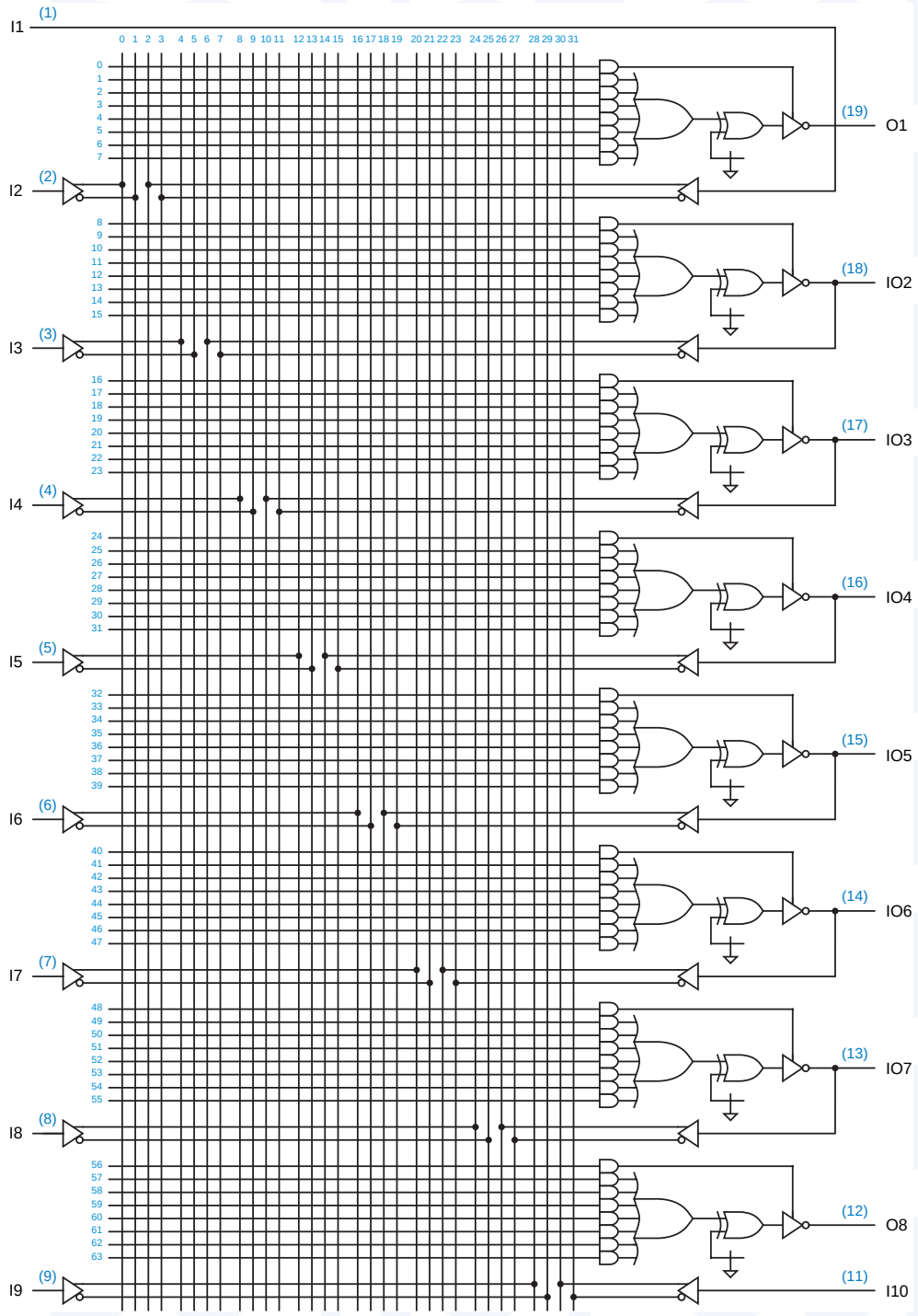


Figure 5-27 Logic diagram of the GAL16V8C.