DIGITAL DEVICES AND LOGIC FAMILIES

ELECTRICAL DEPARTMENT

FOURTH STAGE

INSTRUCTOR: DR. ANAS MQDAD

LECTURE ONE

PROGRAMMABLE LOGIC

OUTLINES

- ❑ Simple Programmable Logic Devices (SPLDs)
- ❑ Complex Programmable Logic Devices (CPLDs)

Simple Programmable Logic Devices (SPLDs)

- Two major types of simple programmable logic devices (SPLDs) are the **PAL** and the GAL. *PAL* stands for programmable array logic, and ***GAL*** stands for generic array logic. Generally, a PAL is one-time programmable (OTP), and a GAL is a type of PAL that is reprogrammable.
- The term *GAL* is a designation originally used by Lattice Semiconductor and later licensed to other manufacturers. The basic structure of both PALs and GALs is a programmable AND array and a fixed OR array, which is a basic sum-of-products architecture.

SPLD: The PAL

- A **PAL** (programmable array logic) consists of a programmable array of AND gates that connects to a fixed array of OR gates.
- PALs are implemented with fuse process technology and are, therefore, one-time programmable (OTP).
- The PAL structure allows any sum-of-products (SOP) logic expression with a defined number of variables to be implemented.

- A simple PAL structure is shown in Figure 1 for two input variables and one output; most PALs have many inputs and many outputs.
- a programmable array is essentially a grid or matrix of conductors that form rows and columns with a programmable link at each cross point.
- Each programmable link, which is a **fuse** in the case of a PAL, is called a **cell**. Each row is connected to the input of an AND gate, and each column is connected to an input variable or its complement.

- By programming the presence or absence of a fuse connection, any combination of input variables or complements can be applied to a AND gate to form any desired product term.

- The AND gates are connected to an OR gate, creating a sum- of-products (SOP) output.

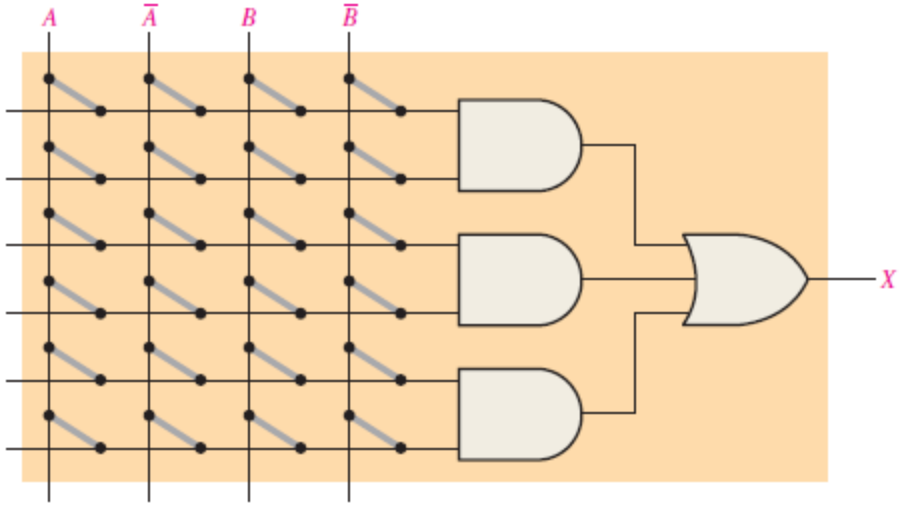


FIGURE 1 Basic AND/OR structure of a PAL.

Implementing a Sum-of-Products Expression

An example of a simple PAL is programmed as shown in Figure 2 so that the product term AB is produced by the top AND gate, $A\overline{B}$ is produced by the middle AND gate, and $\overline{A}\,\overline{B}$ is produced by the bottom AND gate. As you can see, the fuses are left intact to connect the desired variables or their complements to the appropriate AND gate inputs. The fuses are opened where a variable or its complement is not used in a given product term. The final output from the OR gate is the SOP expression,

$$X=AB+A\overline{B}+\overline{A}\,\overline{B}$$

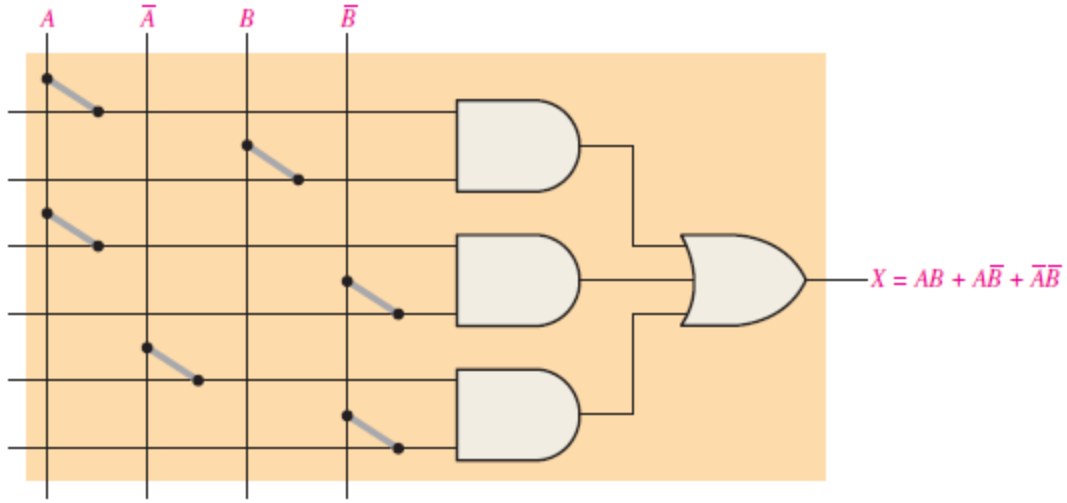


FIGURE 2 PAL implementation of a sum-of-products expression.

SPLD: The GAL

The **GAL** is essentially a PAL that can be reprogrammed. It has the same type of AND/ OR organization that the PAL does. The basic difference is that a GAL uses a reprogrammable process technology, such as EEPROM(E^2CMOS), instead of fuses, as shown in Figure 3.

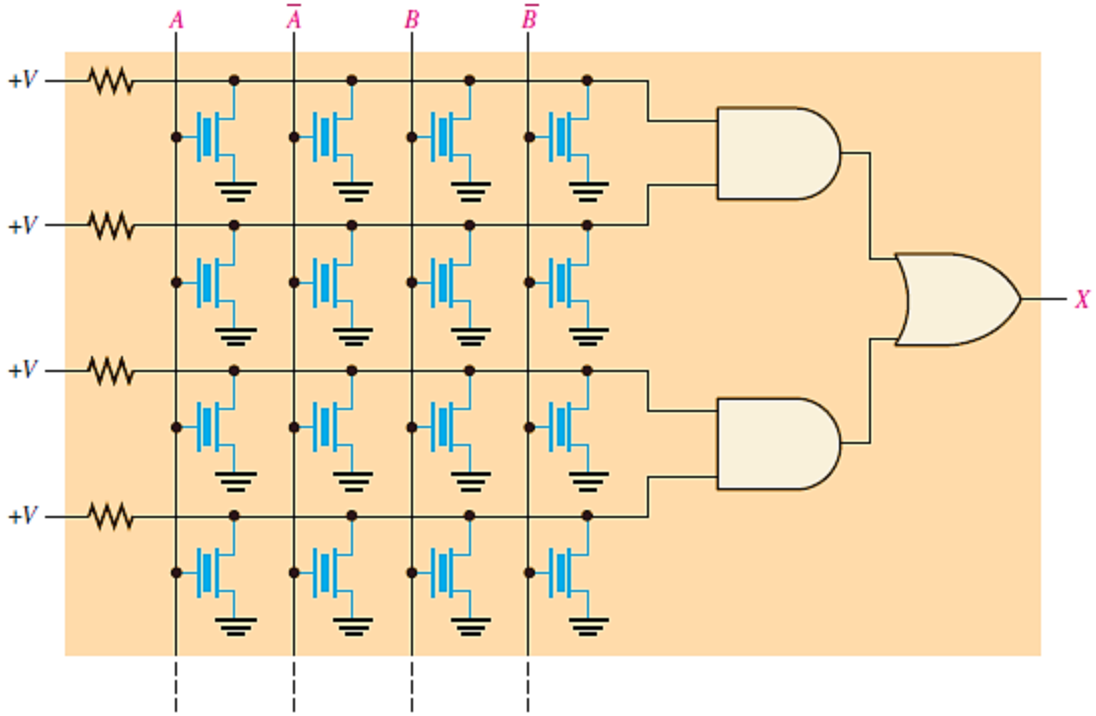


FIGURE 3 Simplified **GAL** array.

Simplified Notation for PAL/GAL Diagrams

- Actual PAL and GAL devices have many AND and OR gates in addition to other elements and are capable of handling many variables and their complements. Most PAL and GAL diagrams that you may see on a data sheet use simplified notation, as illustrated in Figure 4, to keep the schematic from being too complicated.

The input variables to a PAL or GAL are usually buffered to prevent loading by a large number of AND gate inputs to which they are connected.

On the diagram, the triangle symbol represents a buffer that produces both the variable and its complement.

The fixed connections of the input variables and buffers are shown using standard dot notation.

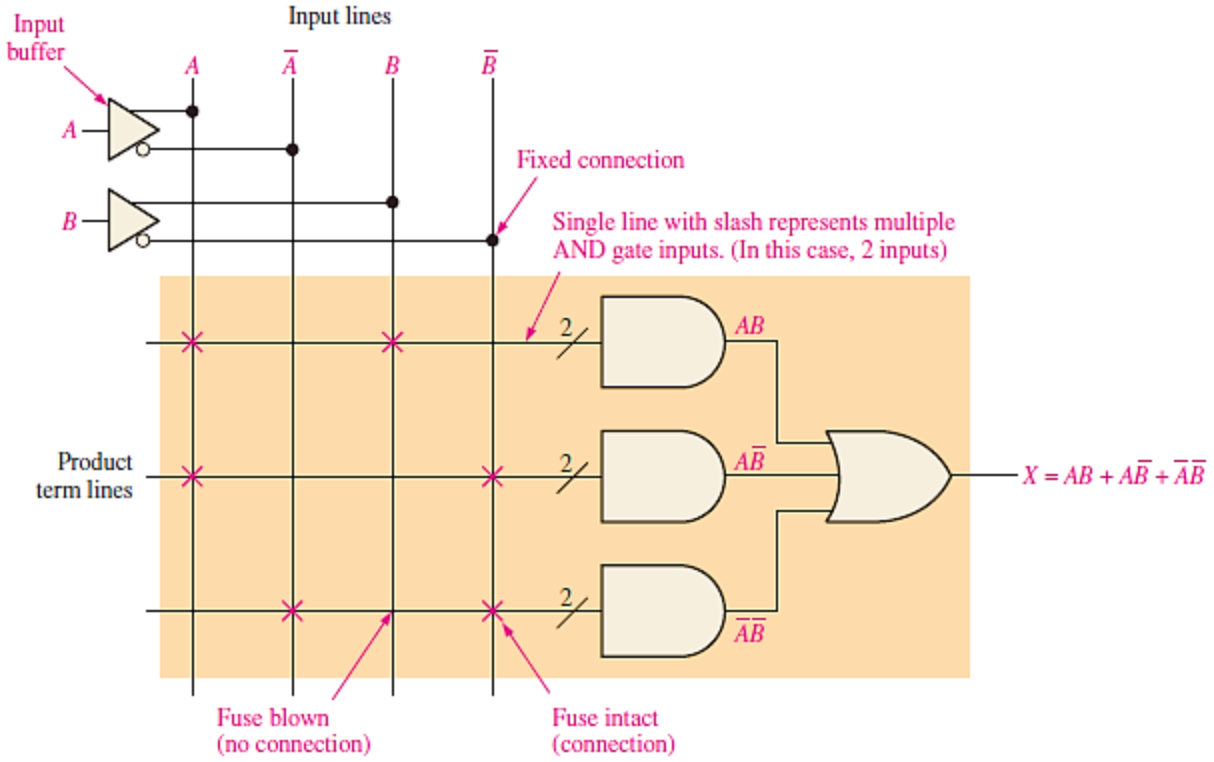


FIGURE 4 A portion of a programmed PAL/GAL.

- PALs and GALs have a large number of programmable interconnection lines, and each AND gate has multiple inputs. Typical PAL and GAL logic diagrams represent a multiple input AND gate with an AND gate symbol having a single input line with a slash and a digit representing the actual number of inputs. Figure 4 illustrates this for the case of 2-input AND gates.

- Programmable links in an array are indicated in a diagram by a red X at the cross point for an intact fuse or other type of link and the absence of an X for an open fuse or other type of link. In Figure 4, the 2-variable logic function $AB + A\overline{B} + \overline{A}\overline{B}$ is programmed.

EXAMPLE

Show how a PAL is programmed for the following 3-variable logic function:

$$X = A\overline{B}C + \overline{A}B\overline{C} + \overline{A}\,\overline{B} + AC$$

Solution

The programmed array is shown in Figure 5. The intact fusible links are indicated by small red Xs. The absence of an X means that the fuse is open.

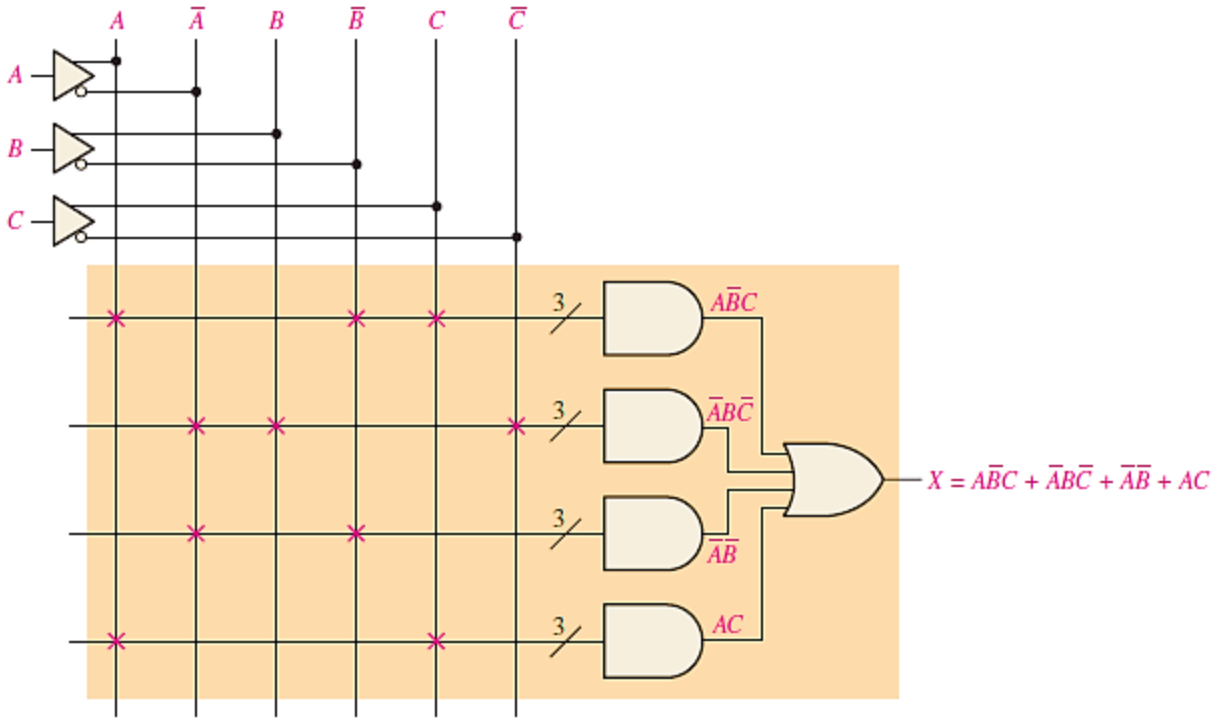


Figure 5

H.W

Write the expression for the output if the fusible links connecting input *A* to the top row and to the bottom row in Figure 5 are open.

PAL/GAL General Block Diagram

A block diagram of a PAL or GAL is shown in Figure 6. Remember, the basic difference is that a GAL has a reprogrammable array and the PAL is one-time programmable.

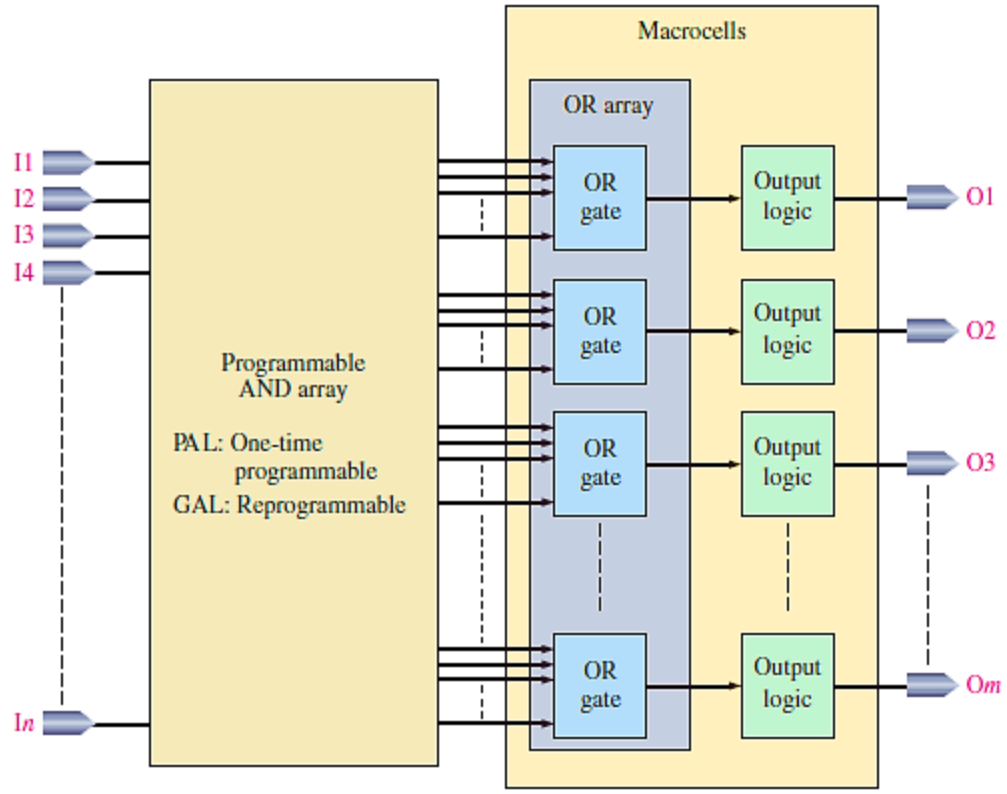


FIGURE 6 General block diagram of a PAL or GAL.

The programmable AND array outputs go to fixed OR gates that are connected to additional output logic. An OR gate combined with its associated output logic is typically called a *macrocell*. The complexity of the macrocell depends on the particular device, and in GALs it is often reprogrammable. Generally, SPLD package configurations range from 20 pins to 28 pins.

Two factors that you can use to help determine whether a certain PAL or GAL is adequate for a given logic design are the number of inputs and outputs and the number of equivalent gates or density. Other parameters to consider are the maximum operating frequency, delay times, and dc supply voltage. Two common types of SPLD are the 16V8 and the 22V10. Various SPLD manufacturers may have different ways of defining density, so you have to use the specified number of equivalent gates with this in mind.

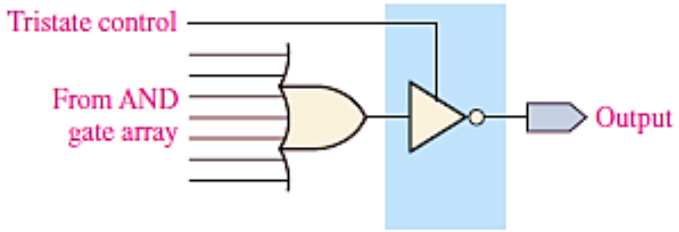
Macrocells

A **macrocell** generally consists of one OR gate and some associated output logic. The macrocells vary in complexity, depending on the particular type of PAL or GAL. A macrocell can be configured for combinational logic, registered logic, or a combination of both. **Registered** logic means that there is a flip-flop in the macrocell to provide for sequential logic functions.

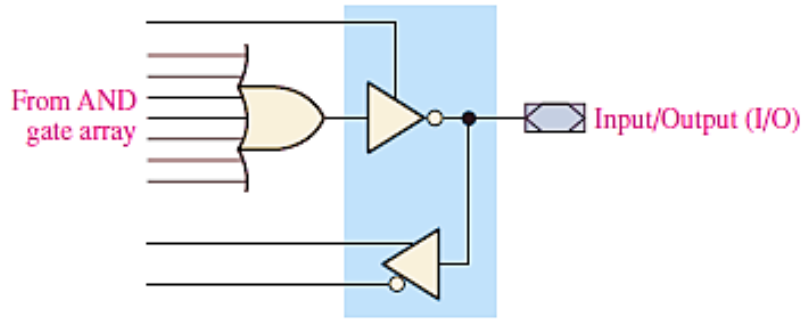
Figure 7 illustrates three basic types of macrocells with combinational logic. **Part (a)** shows a simple macrocell with the OR gate and an inverter with a tristate control that can make the inverter like an open circuit to completely disconnect the output. The output of the tristate inverter can be either LOW, HIGH, or disconnected.

Part (b) is a macrocell that can be either an input or an output. When it is used as an input, the tristate inverter is disconnected, and the input goes to the buffer that is connected to the AND array.

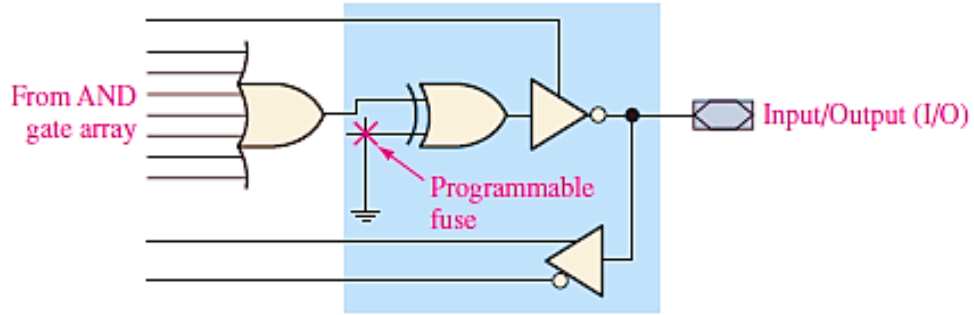
Part (c) is a macrocell that can be programmed to have either an active-HIGH or an active- LOW output, or it can be used as an input. One input to the exclusive-OR (XOR) gate can be programmed to be either HIGH or LOW. When the programmable XOR input is HIGH, the OR gate output is inverted because $0 \oplus 1 = 1$ and $1 \oplus 1 = 0$. Similarly, when the programmable XOR input is LOW, the OR gate output is not inverted because $0 \oplus 0 = 0$ and $1 \oplus 0 = 1$.



(a) Combinational output (active-LOW). An active-HIGH output would be shown without the bubble on the tristate gate symbol.



(b) Combinational input/output (active-LOW)



(c) Programmable polarity output

FIGURE 7 Basic types of PAL/GAL macrocells for combinational logic.

Complex Programmable Logic Devices (CPLDs)

The complex programmable logic device (CPLD) is basically a single device containing multiple SPLDs and providing more capacity for larger logic designs. In this section, the focus is the concepts of traditional CPLD architecture, keeping in mind that CPLDs may vary somewhat in architecture and/or in parameters such as density, process technology, power consumption, supply voltage, and speed.

The CPLD

A **CPLD** (complex programmable logic device) consists basically of multiple SPLD arrays with programmable interconnections. Although the way CPLDs are internally organized varies with the manufacturer, Figure 8 illustrates a generic CPLD. We will refer to each

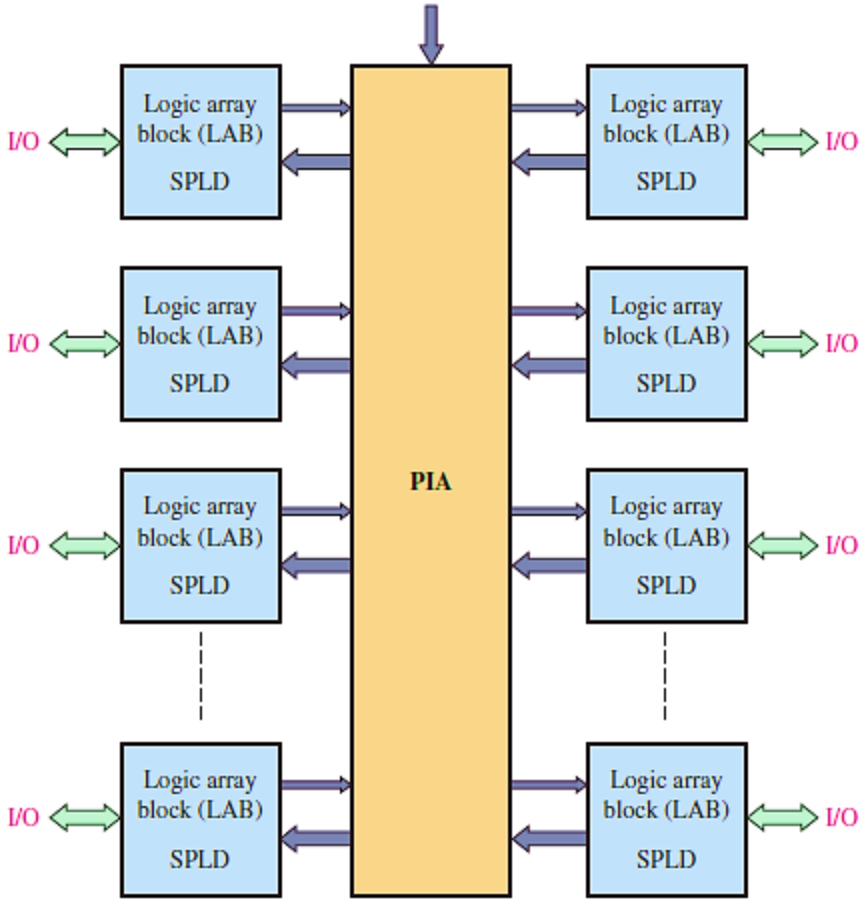


FIGURE 8 Basic block diagram of a generic CPLD.

SPLD array in a CPLD as a **LAB** (logic array block). Other designations are sometimes used, such as *function block, logic block,* or *generic block.* The programmable interconnections are generally called the *PIA* (**programmable interconnect array**) although some manufacturers, such as Xilinx, use the term *AIM* (advanced interconnect matrix) or a similar designation.
The LABs and the interconnections between LABs are programmed using software. A CPLD can be programmed for complex logic functions based on the SOP structure of the individual LABs (actually SPLDs). Inputs can be connected to any of the LABs, and their outputs can be interconnected to any other LABs via the PIA.

Most programmable logic manufacturers make a series of CPLDs that range in density, process technology, power consumption, supply voltage, and speed. Manufacturers usually specify CPLD density in terms of macrocells or logic array blocks. Densities can range from tens of macrocells to over 1500 macrocells in packages with up to several hundred pins. As PLDs become more complex, maximum densities will increase.
Most CPLDs are reprogrammable and use EEPROM or SRAM process technology for the programmable links. Power consumption can range from a few milli watts to a few hundred milliwatts. DC supply voltages are typically from 2.5 V to 5 V, depending on the specific device. Several manufacturers, (for example, Altera, Xilinx, Lattice, and Atmel) produce CPLDs. As you will learn, CPLDs and other programmable logic devices are really a combination of hardware and software.

Classic CPLD Architecture

- The **architecture** of a CPLD is the way in which the internal elements are organized and arranged. The architecture of specific CPLDs is similar to the block diagram of a generic CPLD (shown in Figure 8).
- It has the classic PAL/GAL structure that produces SOP functions. The density ranges from 2 LABs to 16 LABs, depending on the particular device in the series.
- Remember, a LAB is roughly equivalent to one SPLD, and package sizes for CPLDs vary from 44 pins to 208 pins. Typically, a series of CPLDs uses the EEPROM-based process technology. In-system programmable (ISP) versions use the JTAG standard interface.
- Figure 9 shows a general block diagram of a typical CPLD. Four LABs are shown, but there can be up to sixteen, depending on the particular device in a series. Each ofthe four LABs consists of sixteen macrocells, and multiple LABs are linked together via the PIA, which is a programmable global (goes to all LABs) bus structure to which the general-purpose inputs, the I/Os, and the macrocells are connected.

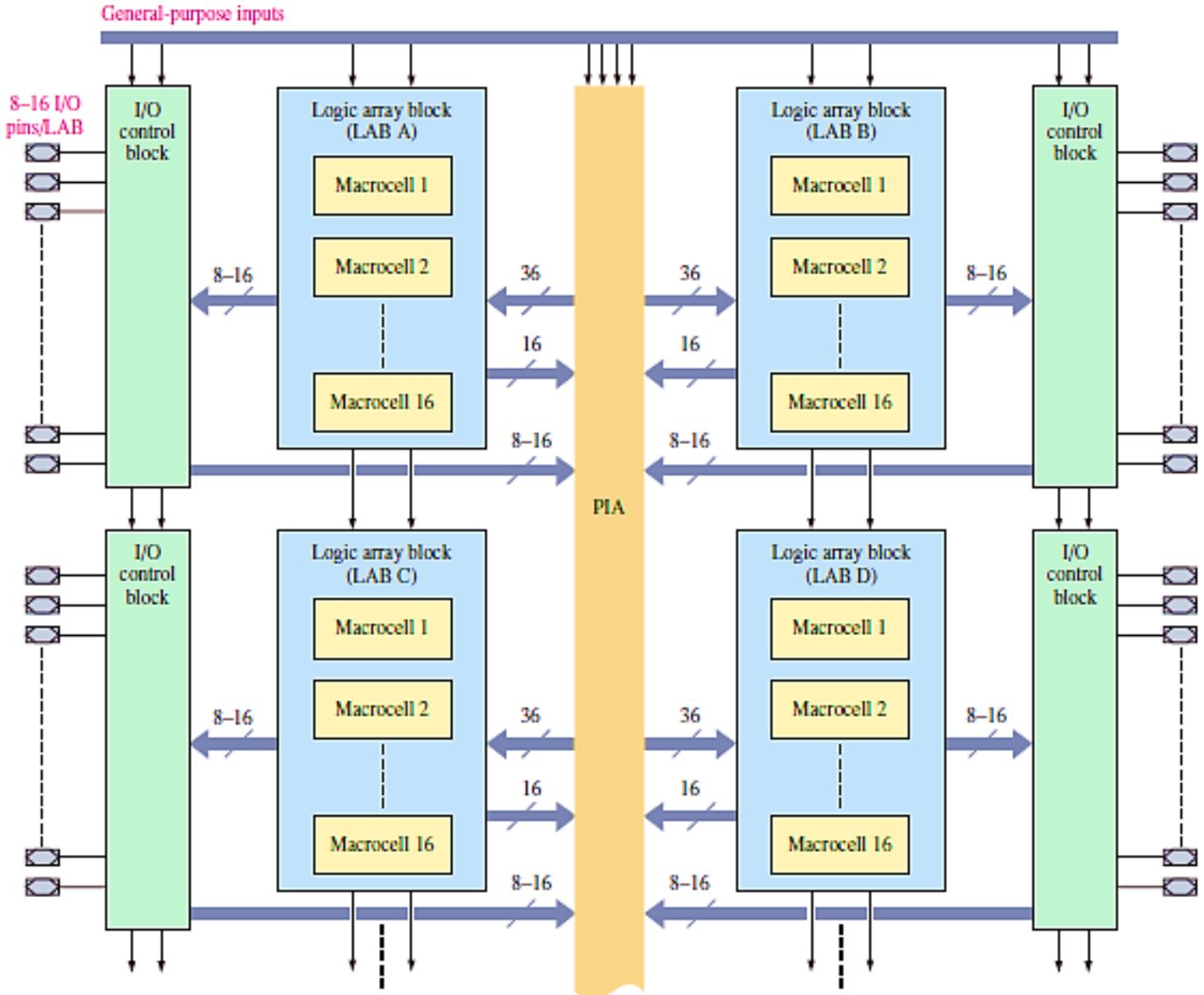


FIGURE 9 Basic block diagram of a typical CPLD.

The Macrocell

A simplified diagram of a typical macrocell is shown in Figure 10. The macrocell contains a small programmable AND array with five AND gates, an OR gate, a product-term selection matrix for connecting the AND gate outputs to the OR gate, and associated logic that can be programmed for input, combinational logic output, or registered output.

This macrocell differs somewhat from the macrocell discussed before in relation to SPLDs because it contains a portion of the programmable AND array and a product-term selection matrix. As shown in Figure 10, five AND gates feed product terms from the PIA into the product-term selection matrix. The product term from the bottom AND gate can be fed back inverted into the programmable array as a shared expander for use by other macrocells. The parallel expander inputs allow borrowing of unused product terms from other macrocells to expand an SOP expression. The product-term selectionnmatrix is an array of programmable connections that is used to connect selected outputs from the AND array and from the expander inputs to the OR gate.

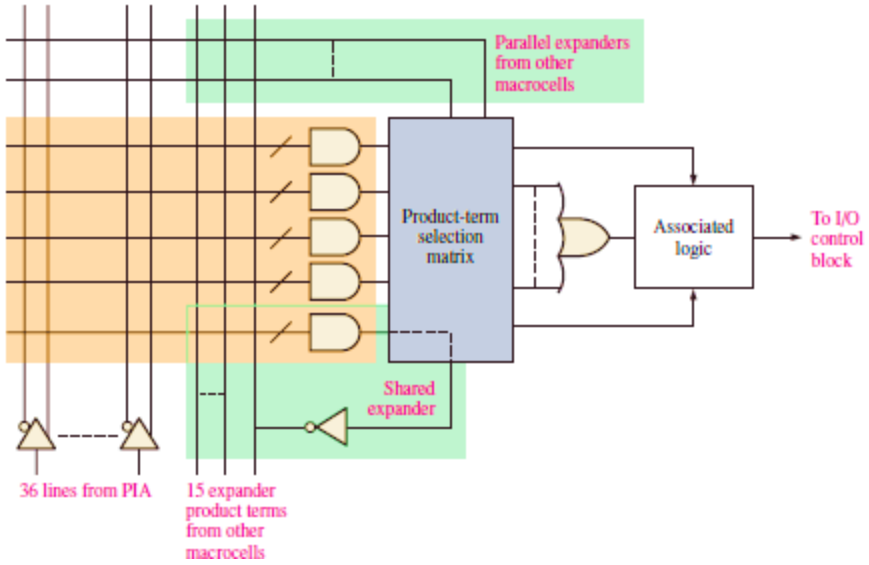


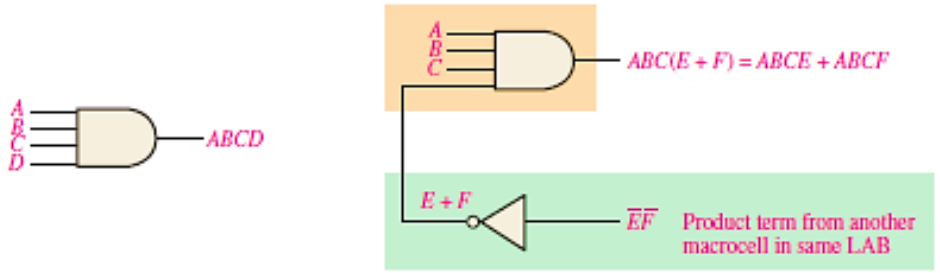
FIGURE 10 Simplified diagram of a macrocell in a typical CPLD.

Shared Expanders

A complemented product term that can be used to increase the number of product terms in an SOP expression is available from each macrocell in a LAB. Figure 11 illustrates how a shared expander term from another macrocell can be used to create additional product terms. In this case, each of the five AND gates in a macrocell array is limited to four inputs

and, therefore, can produce up to a 4-variable product term, as illustrated in part (a). Figure 11(b) shows the expansion to two product terms.

Each macrocell can produce up to five product terms generated from its AND array. If a macrocell needs more than five product terms for its SOP output, it can use an expander term from another macrocell. Suppose that a design requires an SOP expression that contains six product terms.



(a) A 4-input AND array gate can produce one 4-variable product term.

(b) AND gate is expanded to produce two product terms.

FIGURE 11 Example of how a shared expander can be used in a macrocell to increase the number of product terms.

Figure 12 shows how a product term from another macrocell can be used to increase an SOP output. Macrocell 2, which is underutilized, generates a shared expander term ($E + F$) that connects to the fifth AND gate in macrocell 1 to produce an SOP expression with six product terms.

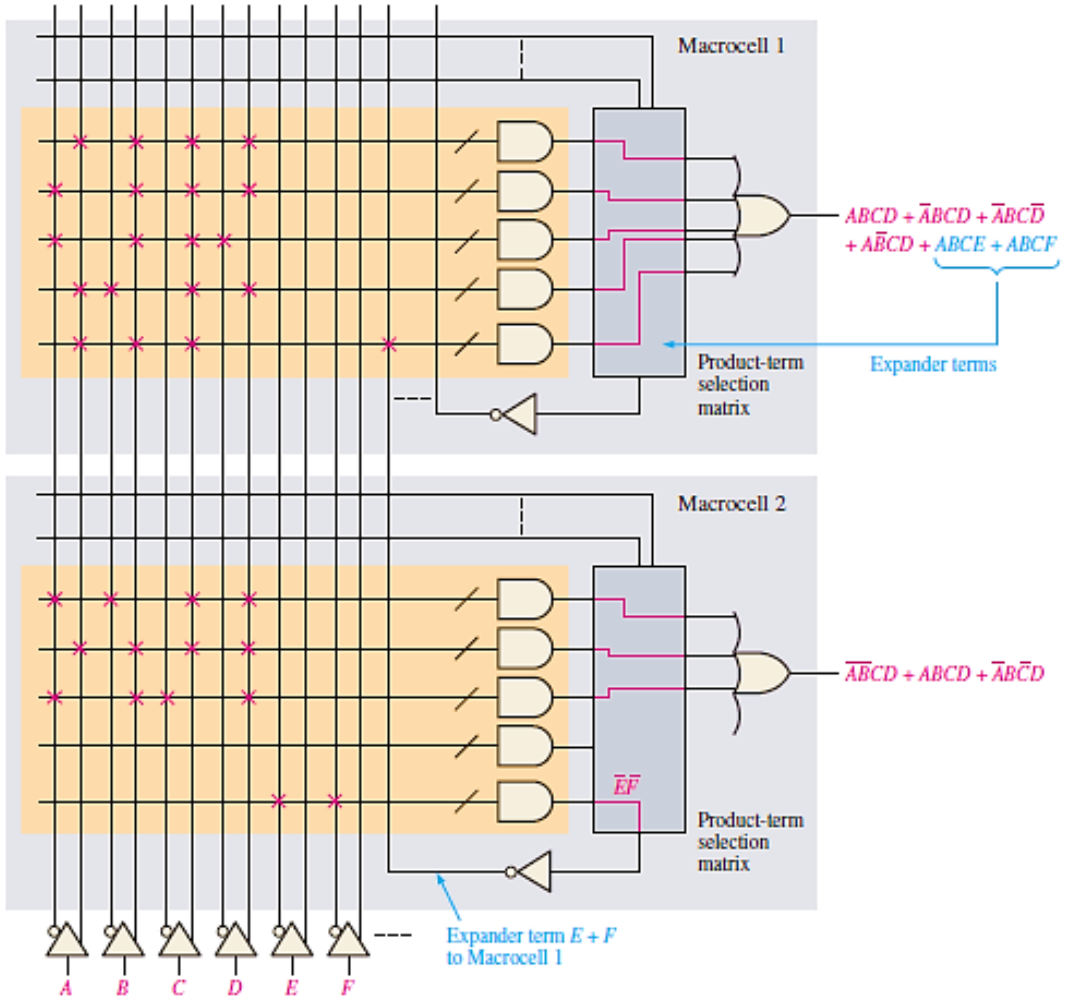


FIGURE 12 Simplified illustration of using a shared expander term from another macrocell to increase an SOP expression. The red Xs and lines represent the connections produced in the hardware by the software compiler running the programmed design.

Parallel Expanders

Another way to increase the number of product terms for a macrocell is by using parallel expanders in which additional product terms are ORed with the terms generated by a macrocell instead of being combined in the AND array, as in the shared expander. A given macrocell can borrow unused product terms from neighboring macrocells. The basic concept is illustrated in Figure 13 where a simplified circuit that can produce two product terms borrows three additional product terms.

Figure 14 shows how one macrocell can borrow parallel expander terms from another macrocell to increase the SOP output. Macrocell 2 uses three product terms from macrocell 1 to produce an eight-term SOP expression.

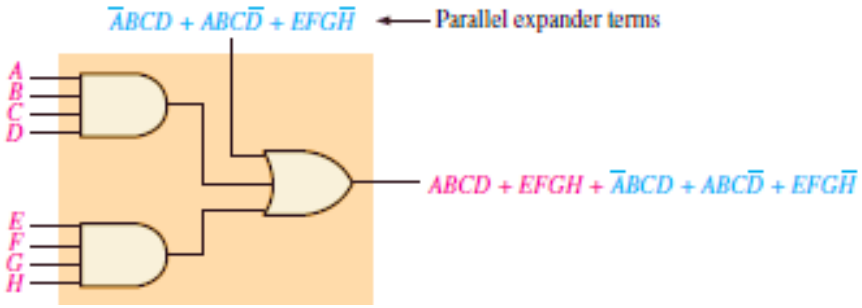


FIGURE 13 Basic concept of the parallel expander.

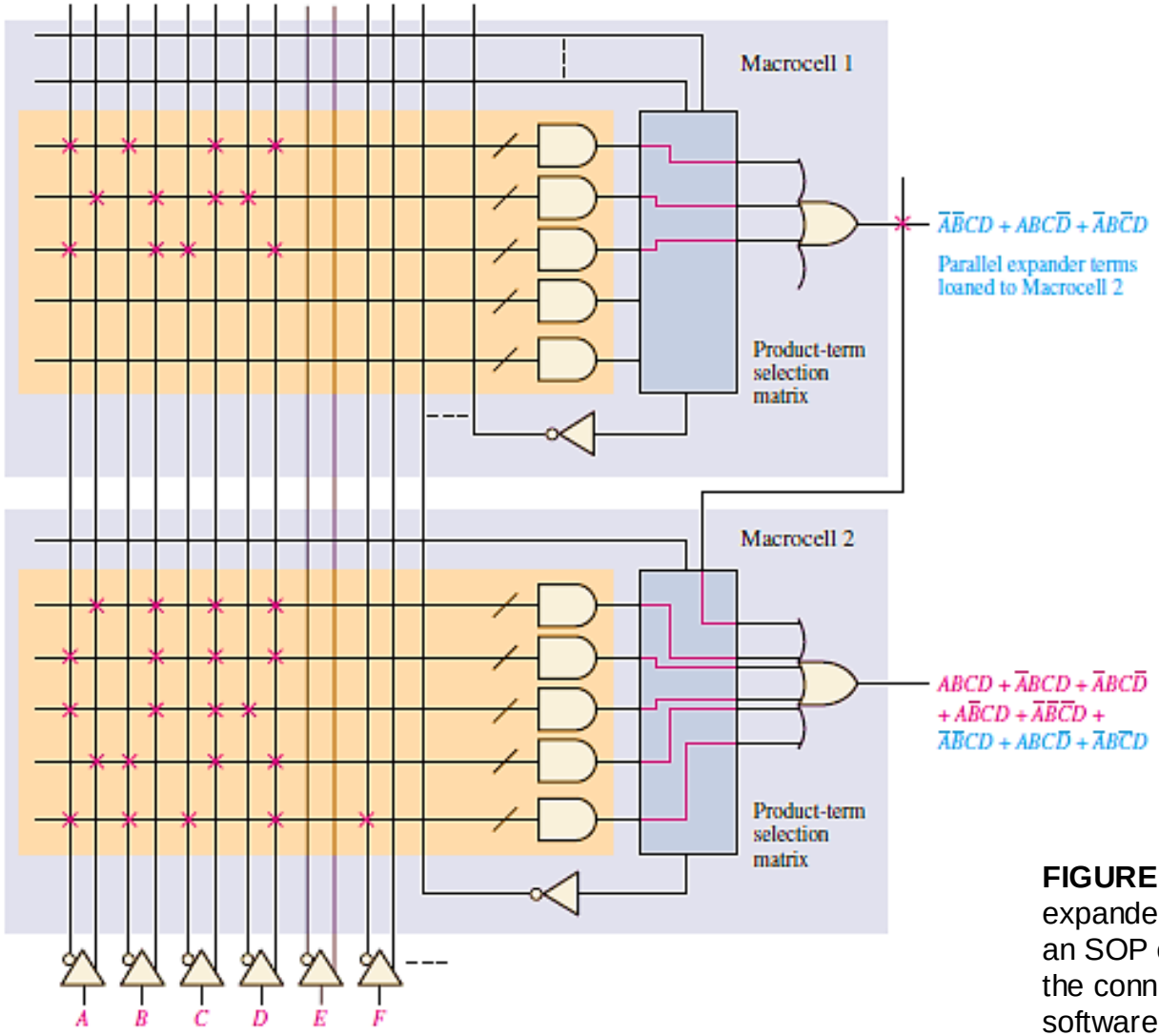


FIGURE 14 Simplified illustration of using parallel expander terms from another macrocell to increase an SOP expression. The red Xs and lines represent the connections produced in the hardware by the software complier running the programmed design.

Thank You