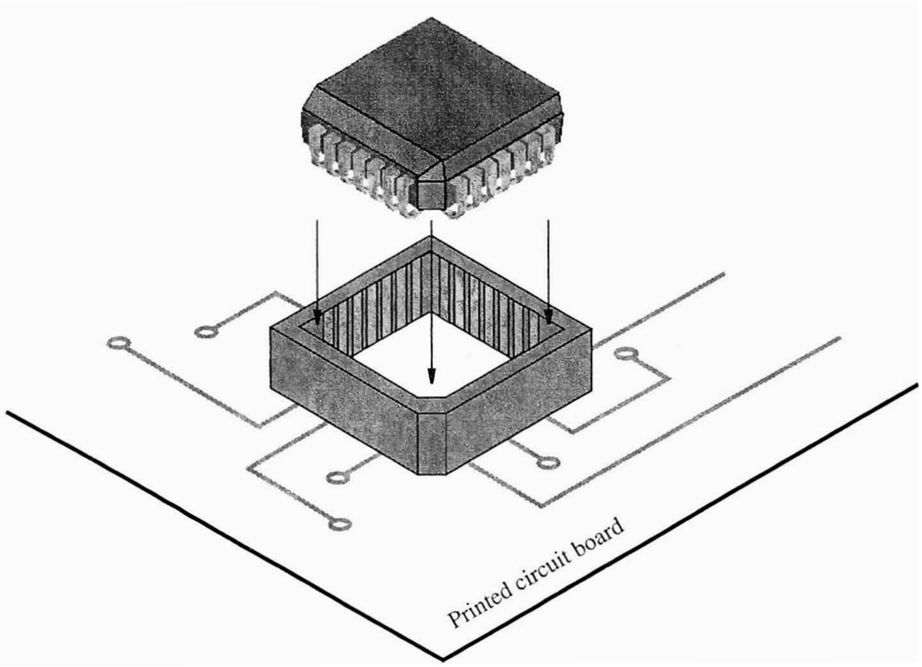


Figure 3.31 A PLCC package with socket.

3.6.4 Complex Programmable Logic Devices (CPLDs)

PLAs and PALs are useful for implementing a wide variety of small digital circuits. Each device can be used to implement circuits that do not require more than the number of inputs, product terms, and outputs that are provided in the particular chip. These chips are limited to fairly modest sizes, typically supporting a combined number of inputs plus outputs of not more than 32. For implementation of circuits that require more inputs and outputs, either multiple PLAs or PALs can be employed or else a more sophisticated type of chip, called a *complex programmable logic device (CPLD)*, can be used.

A CPLD comprises multiple circuit blocks on a single chip, with internal wiring resources to connect the circuit blocks. Each circuit block is similar to a PLA or a PAL; we will refer to the circuit blocks as *PAL-like blocks*. An example of a CPLD is given in Figure 3.32. It includes four PAL-like blocks that are connected to a set of *interconnection wires*. Each PAL-like block is also connected to a subcircuit labeled *I/O block*, which is attached to a number of the chip's input and output pins.

Figure 3.33 shows an example of the wiring structure and the connections to a PAL-like block in a CPLD. The PAL-like block includes 3 macrocells (real CPLDs typically have about 16 macrocells in a PAL-like block), each consisting of a four-input OR gate (real CPLDs usually provide between 5 and 20 inputs to each OR gate). The OR-gate output is connected to another type of logic gate that we have not yet introduced. It is called an Exclusive-OR (XOR) gate. We discuss XOR gates in section 3.9.1. The behavior of an XOR gate is the same as for an OR gate except that if both of the inputs are 1, the XOR gate

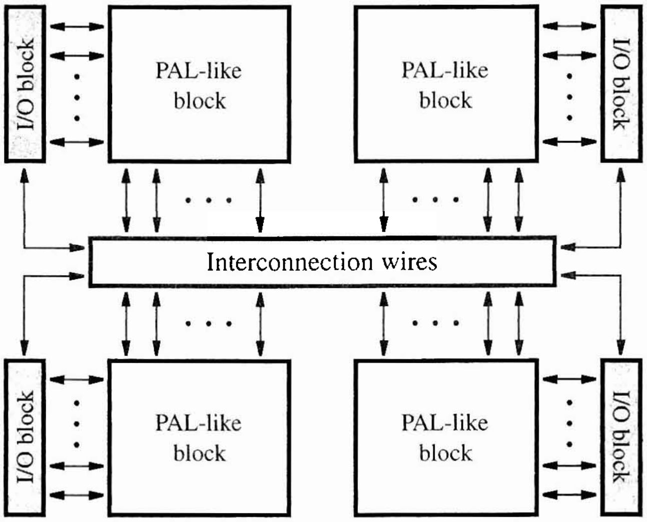


Figure 3.32 Structure of a complex programmable logic device (CPLD).

produces a 0. One input to the XOR gate in Figure 3.33 can be programmably connected to 1 or 0; if 1, then the XOR gate complements the OR-gate output, and if 0, then the XOR gate has no effect. The macrocell also includes a flip-flop, a multiplexer, and a tri-state buffer. As we mentioned in the discussion for Figure 3.29, the flip-flop is used to store the output value produced by the OR gate. Each tri-state buffer (see section 3.8.8) is connected to a pin on the CPLD package. The tri-state buffer acts as a switch that allows each pin to be used either as an output from the CPLD or as an input. To use a pin as an output, the corresponding tri-state buffer is enabled, acting as a switch that is turned on. If the pin is to be used as an input, then the tri-state buffer is disabled, acting as a switch that is turned off. In this case an external source can drive a signal onto the pin, which can be connected to other macrocells using the interconnection wiring.

The interconnection wiring contains programmable switches that are used to connect the PAL-like blocks. Each of the horizontal wires can be connected to some of the vertical wires that it crosses, but not to all of them. Extensive research has been done to decide how many switches should be provided for connections between the wires. The number of switches is chosen to provide sufficient flexibility for typical circuits without wasting many switches in practice. One detail to note is that when a pin is used as an input, the macrocell associated with that pin cannot be used and is therefore wasted. Some CPLDs include additional connections between the macrocells and the interconnection wiring that avoids wasting macrocells in such situations.

Commercial CPLDs range in size from only 2 PAL-like blocks to more than 100 PAL-like blocks. They are available in a variety of packages, including the PLCC package that is shown in Figure 3.31. Figure 3.34*a* shows another type of package used to house CPLD chips, called a *quad flat pack* (QFP). Like a PLCC package, the QFP package has pins on all

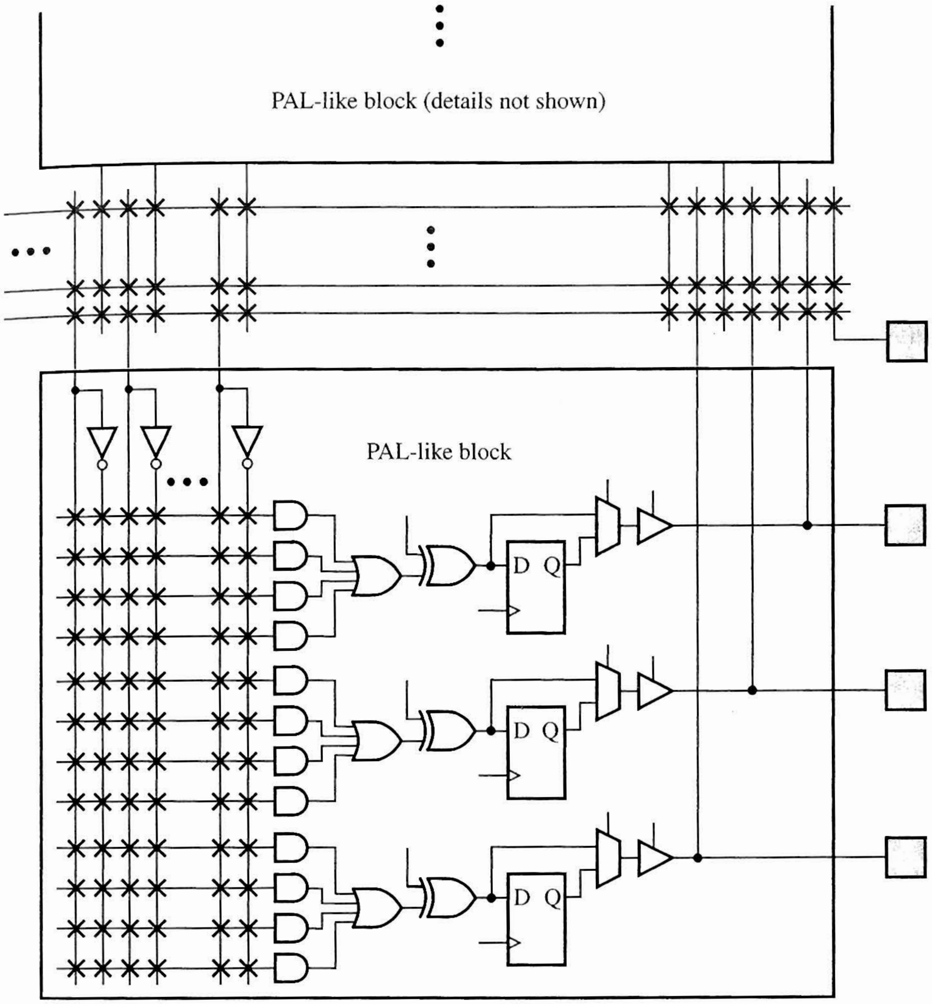
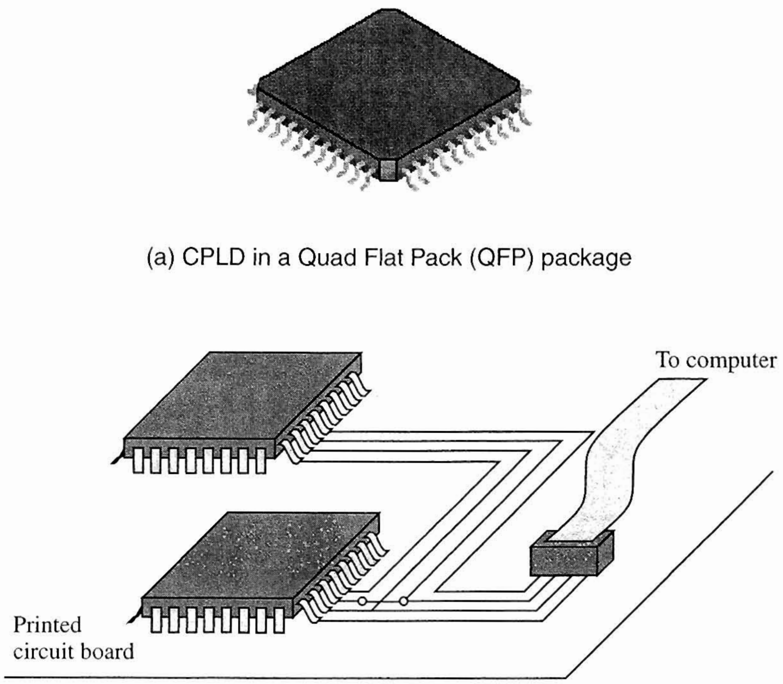


Figure 3.33 A section of the CPLD in Figure 3.32.

four sides, but whereas the PLCC's pins wrap around the edges of the package, the QFP's pins extend outward from the package, with a downward-curving shape. The QFP's pins are much thinner than those on a PLCC, which means that the package can support a larger number of pins; QFPs are available with more than 200 pins, whereas PLCCs are limited to fewer than 100 pins.

Most CPLDs contain the same type of programmable switches that are used in SPLDs, which are described in section 3.10. Programming of the switches may be accomplished using the same technique described in section 3.6.3, in which the chip is placed into a special-purpose programming unit. However, this programming method is rather inconvenient for large CPLDs for two reasons. First, large CPLDs may have more than 200 pins on the chip



(a) CPLD in a Quad Flat Pack (QFP) package

(b) JTAG programming

Figure 3.34 CPLD packaging and programming.

package, and these pins are often fragile and easily bent. Second, to be programmed in a programming unit, a socket is required to hold the chip. Sockets for large QFP packages are very expensive; they sometimes cost more than the CPLD device itself. For these reasons, CPLD devices usually support the ISP technique. A small connector is included on the PCB that houses the CPLD, and a cable is connected between that connector and a computer system. The CPLD is programmed by transferring the programming information generated by a CAD system through the cable, from the computer into the CPLD. The circuitry on the CPLD that allows this type of programming has been standardized by the IEEE and is usually called a *JTAG port*. It uses four wires to transfer information between the computer and the device being programmed. The term *JTAG* stands for Joint Test Action Group. Figure 3.34*b* illustrates the use of a JTAG port for programming two CPLDs on a circuit board. The CPLDs are connected together so that both can be programmed using the same connection to the computer system. Once a CPLD is programmed, it retains the programmed state permanently, even when the power supply for the chip is turned off. This property is called *nonvolatile* programming.

CPLDs are used for the implementation of many types of digital circuits. In industrial designs that employ some type of PLD device, CPLDs are used often, while SPLDs are becoming less common. A number of companies offer competing CPLDs. Appendix E lists,

in Table
locators
provide
detail ir

3.6.5

The typ
plemen
small ar
erately
reasons
the amc
to quan
gates ar
the tota
measur

Us
because
used is
has eig
large C
gates.

By
larger c
A *field-*
implem
and CP
logic bl
is illust
blocks
The log
are org
logic b
the logi
prograr
the logi
that are
as a ver
I/O blo
and wir

FP
gates ir
describ

in Table E.2, the names of the major companies involved and shows the companies” WWW locators. The reader is encouraged to examine the product information that each company provides on its Web pages. An example of a popular commercial CPLD is described in detail in Appendix E.

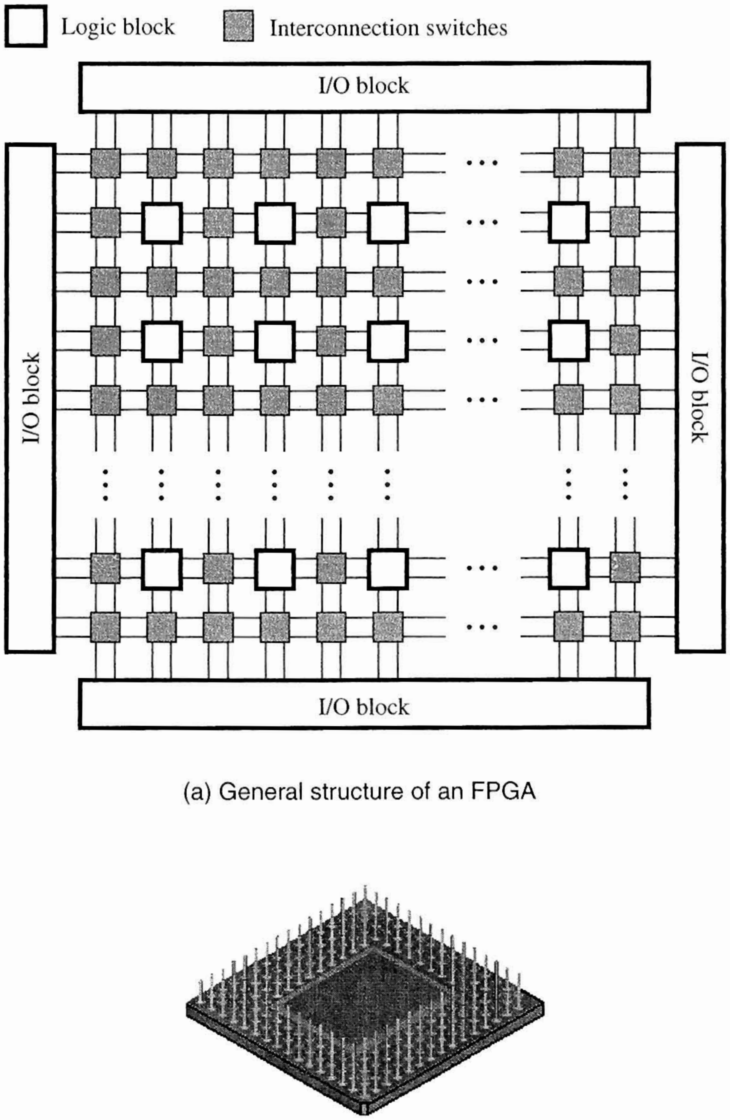
3.6.5 Field-Programmable Gate Arrays

The types of chips described above, 7400 series, SPLDs, and CPLDs, are useful for implementation of a wide range of logic circuits. Except for CPLDs, these devices are rather small and are suitable only for relatively simple applications. Even for CPLDs, only moderately large logic circuits can be accommodated in a single chip. For cost and performance reasons, it is prudent to implement a desired logic circuit using as few chips as possible, so the amount of circuitry on a given chip and its functional capability are important. One way to quantify a circuit's *size* is to assume that the circuit is to be built using only simple logic gates and then estimate how many of these gates are needed. A commonly used measure is the total number of two-input NAND gates that would be needed to build the circuit; this measure is often called the number of *equivalent gates*.

Using the equivalent-gates metric, the size of a 7400-series chip is simple to measure because each chip contains only simple gates. For SPLDs and CPLDs the typical measure used is that each macrocell represents about 20 equivalent gates. Thus a typical PAL that has eight macrocells can accommodate a circuit that needs up to about 160 gates, and a large CPLD that has 500 macrocells can implement circuits of up to about 10,000 equivalent gates.

By modern standards, a logic circuit with 10,000 gates is not large. To implement larger circuits, it is convenient to use a different type of chip that has a larger logic capacity. A *field-programmable gate array (FPGA)* is a programmable logic device that supports implementation of relatively large logic circuits. FPGAs are quite different from SPLDs and CPLDs because FPGAs do not contain AND or OR planes. Instead, FPGAs provide *logic blocks* for implementation of the required functions. The general structure of an FPGA is illustrated in Figure 3.35*a*. It contains three main types of resources: logic blocks, I/O blocks for connecting to the pins of the package, and interconnection wires and switches. The logic blocks are arranged in a two-dimensional array, and the interconnection wires are organized as horizontal and vertical *routing channels* between rows and columns of logic blocks. The routing channels contain wires and programmable switches that allow the logic blocks to be interconnected in many ways. Figure 3.35*a* shows two locations for programmable switches; the blue boxes adjacent to logic blocks hold switches that connect the logic block input and output terminals to the interconnection wires, and the blue boxes that are diagonally between logic blocks connect one interconnection wire to another (such as a vertical wire to a horizontal wire). Programmable connections also exist between the I/O blocks and the interconnection wires. The actual number of programmable switches and wires in an FPGA varies in commercially available chips.

FPGAs can be used to implement logic circuits of more than a million equivalent gates in size. Some examples of commercial FPGA products, from Altera and Xilinx, are described in Appendix E. FPGA chips are available in a variety of packages, including the

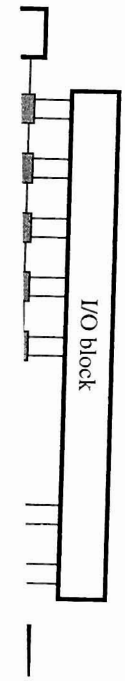


(a) General structure of an FPGA

(b) Pin grid array (PGA) package (bottom view)

Figure 3.35 A field-programmable gate array (FPGA).

PLCC and QFP packages described earlier. Figure 3.35*b* depicts another type of package, called a *pin grid array (PGA)*. A PGA package may have up to a few hundred pins in total, which extend straight outward from the bottom of the package, in a grid pattern. Yet another packaging technology that has emerged is known as the *ball grid array (BGA)*. The BGA is similar to the PGA except that the pins are small round balls, instead of posts.

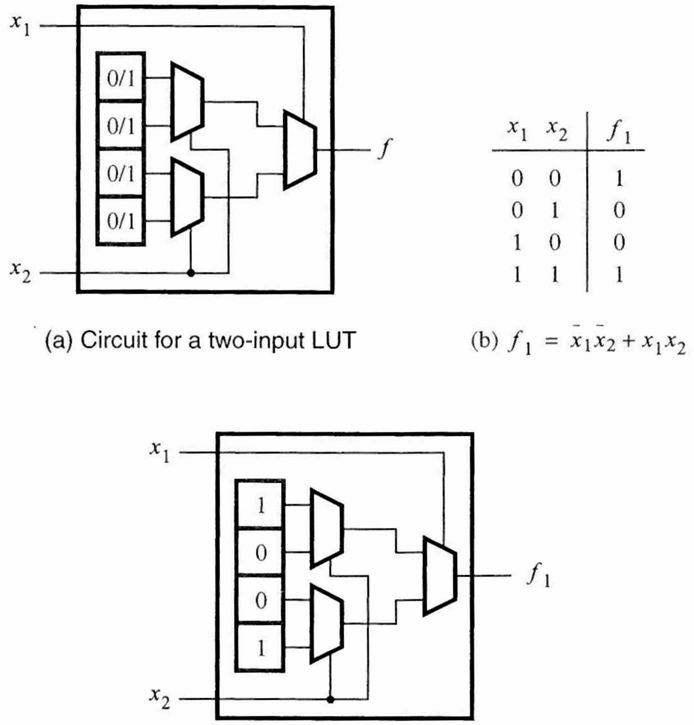


type of package,
hundred pins in
grid pattern. Yet
d array (BGA).
instead of posts.

The advantage of BGA packages is that the pins are very small; hence more pins can be provided on a relatively small package.

Each logic block in an FPGA typically has a small number of inputs and outputs. A variety of FPGA products are on the market, featuring different types of logic blocks. The most commonly used logic block is a *lookup table (LUT)*, which contains *storage cells* that are used to implement a small logic function. Each cell is capable of holding a single logic value, either 0 or 1. The stored value is produced as the output of the storage cell. LUTs of various *sizes* may be created, where the size is defined by the number of inputs. Figure 3.36*a* shows the structure of a small LUT. It has two inputs, x_1 and x_2, and one output, f. It is capable of implementing any logic function of two variables. Because a two-variable truth table has four rows, this LUT has four storage cells. One cell corresponds to the output value in each row of the truth table. The input variables x_1 and x_2 are used as the select inputs of three multiplexers, which, depending on the valuation of x_1 and x_2, select the content of one of the four storage cells as the output of the LUT. We introduced multiplexers in section 2.8.2 and will discuss storage cells in Chapter 10.

To see how a logic function can be realized in the two-input LUT, consider the truth table in Figure 3.36*b*. The function f_1 from this table can be stored in the LUT as illustrated in



(a) Circuit for a two-input LUT

x_1	x_2	f_1
0	0	1
0	1	0
1	0	0
1	1	1

(b) $f_1 = \bar{x}_1\bar{x}_2 + x_1x_2$

(c) Storage cell contents in the LUT

Figure 3.36 A two-input lookup table (LUT).

Figure 3.36*c*. The arrangement of multiplexers in the LUT correctly realizes the function f_1. When $x_1 = x_2 = 0$, the output of the LUT is driven by the top storage cell, which represents the entry in the truth table for $x_1x_2 = 00$. Similarly, for all valuations of x_1 and x_2, the logic value stored in the storage cell corresponding to the entry in the truth table chosen by the particular valuation appears on the LUT output. Providing access to the contents of storage cells is only one way in which multiplexers can be used to implement logic functions. A detailed presentation of the applications of multiplexers is given in Chapter 6.

Figure 3.37 shows a three-input LUT. It has eight storage cells because a three-variable truth table has eight rows. In commercial FPGA chips, LUTs usually have either four or five inputs, which require 16 and 32 storage cells, respectively. In Figure 3.29 we showed that PALs usually have extra circuitry included with their AND-OR gates. The same is true for FPGAs, which usually have extra circuitry, besides a LUT, in each logic block. Figure 3.38 shows how a flip-flop may be included in an FPGA logic block. As discussed for

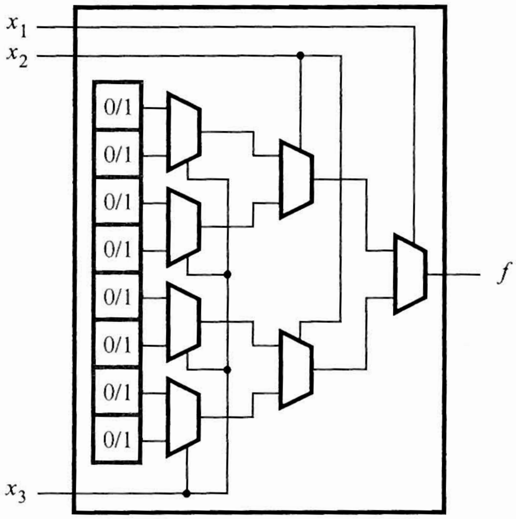


Figure 3.37 A three-input LUT.

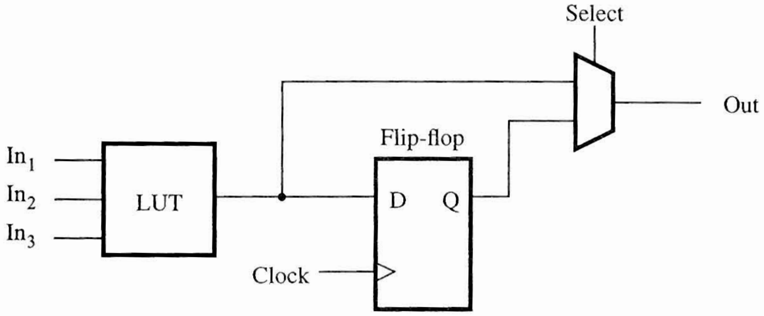


Figure 3.38 Inclusion of a flip-flop in an FPGA logic block.

Figure 3.29, the flip-flop is used to store the value of its D input under control of its *clock* input. Examples of logic blocks in commercial FPGAs are presented in Appendix E.

For a logic circuit to be realized in an FPGA, each logic function in the circuit must be small enough to fit within a single logic block. In practice, a user's circuit is automatically translated into the required form by using CAD tools (see Chapter 12). When a circuit is implemented in an FPGA, the logic blocks are programmed to realize the necessary functions and the routing channels are programmed to make the required interconnections between logic blocks. FPGAs are configured by using the ISP method, which we explained in section 3.6.4. The storage cells in the LUTs in an FPGA are *volatile*, which means that they lose their stored contents whenever the power supply for the chip is turned off. Hence the FPGA has to be programmed every time power is applied. Often a small memory chip that holds its data permanently, called a *programmable read-only memory (PROM)*, is included on the circuit board that houses the FPGA. The storage cells in the FPGA are loaded automatically from the PROM when power is applied to the chips.

A small FPGA that has been programmed to implement a circuit is depicted in Figure 3.39. The FPGA has two-input LUTs, and there are four wires in each routing channel. The figure shows the programmed states of both the logic blocks and wiring switches in a section of the FPGA. Programmable wiring switches are indicated by an X. Each switch shown in blue is turned on and makes a connection between a horizontal and vertical wire.

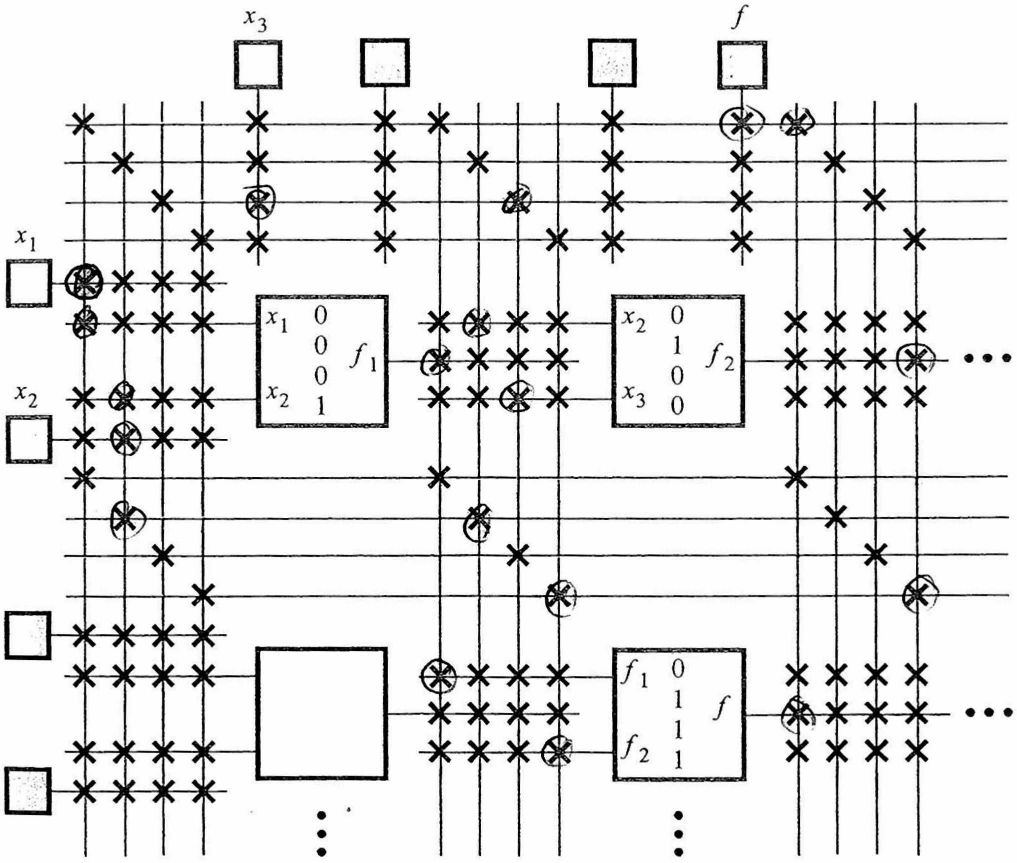


Figure 3.39 A section of a programmed FPGA.

The switches shown in black are turned off. We describe how the switches are implemented by using transistors in section 3.10.1. The truth tables programmed into the logic blocks in the top row of the FPGA correspond to the functions $f_1 = x_1x_2$ and $f_2 = \overline{x}_2x_3$. The logic block in the bottom right of the figure is programmed to produce $f = f_1 + f_2 = x_1x_2 + \overline{x}_2x_3$.

3.6.6 Using CAD Tools to Implement Circuits in CPLDs and FPGAs

In section 2.9 we suggested the reader should work through Tutorial 1, in Appendix B, to gain some experience using real CAD tools. Tutorial 1 covers the steps of design entry and functional simulation. Now that we have discussed some of the details of the implementation of circuits in chips, the reader may wish to experiment further with the CAD tools. In Tutorials 2 and 3 (Appendices C and D) we show how circuits designed with CAD tools can be implemented in CPLD and FPGA chips.

3.6.7 Applications of CPLDs and FPGAs

CPLDs and FPGAs are used today in many diverse applications, such as consumer products like DVD players and high-end television sets, controller circuits for automobile factories and test equipment, Internet routers and high-speed network switches, and computer equipment like large tape and disk storage systems.

In a given design situation a CPLD may be chosen when the needed circuit is not very large, or when the device has to perform its function immediately upon application of power to the circuit. FPGAs are not a good choice for this latter case because, as we mentioned before, they are configured by volatile storage elements that lose their stored contents when the power is turned off. This property results in a delay before the FPGA chip can perform its function when turned on.

FPGAs are suitable for implementation of circuits over a large range of size, from about 1000 to more than a million equivalent logic gates. In addition to size a designer will consider other criteria, such as the needed speed of operation of a circuit, power dissipation constraints, and the cost of the chips. When FPGAs do not meet one or more of the requirements, the user may choose to create a custom-manufactured chip as described below.

3.7 Custom Chips, Standard Cells, and Gate Arrays

The key factor that limits the size of a circuit that can be accommodated in a PLD is the existence of programmable switches. Although these switches provide the important benefit of user programmability, they consume a significant amount of space on the chip, which leads to increased cost. They also result in a reduction in the speed of operation of circuits, and an increase in power consumption. In this section we will introduce some integrated circuit technologies that do not contain programmable switches.