are implemented within the logic cells. The cells are arranged on a grid of routing resources that can make connections between arbitrary cells to build logic paths as shown in Fig. 11.7. Depending on the FPGA type, special-purpose structures are placed into the array. Most often, these are configurable RAM blocks and clock distribution elements. Around the periphery of the chip are the I/O cells, which commonly contain one or more flops to enable high-performance synchronous interfaces. Locating flops within I/O cells improves timing characteristics by minimizing the distance, and hence the delay between each flop and its associated pin. Unlike CPLDs, most FPGAs are based on SRAM technology, making their configurations volatile. A typical FPGA must be reprogrammed each time power is applied to a system. Major vendors of FPGAs include Actel, Altera, Atmel, Lattice, QuickLogic, and Xilinx.

Very high logic densities are achieved by scaling the size of the two-dimensional logic cell array. The primary limiting factor in FPGA performance becomes routing because of the nondeterministic nature of a multipath grid interconnect system. Paths between logic cells can take multiple routes, some of which may provide identical propagation delays. However, routing resources are finite, and conflicts quickly arise between competing paths for the same routing channels. As with CPLDs, FPGA vendors provide proprietary software tools to convert a netlist into a final programming image. Depending on the complexity of the design (e.g., speed and density), routing an FPGA can take a few minutes or many hours. Unlike a CPLD with deterministic interconnection resources, FPGA timing can vary dramatically, depending on the quality of the logic placement. Large, fast designs require iterative routing and placement algorithms.

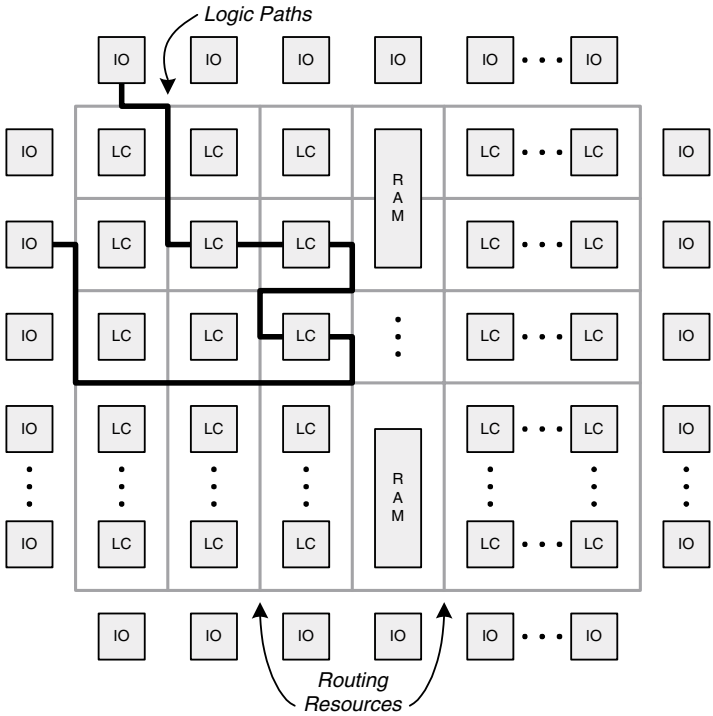

FIGURE 11.7 FPGA logic cell array.

Human intervention can be critical to the successful routing and timing of a complex FPGA design. *Floorplanning* is the process by which an engineer manually partitions logic into groups and then explicitly places these groups into sections of the logic cell array. Manually locating large portions of the logic restricts the routing software to optimizing placement of logic within those boundaries and reduces the number of permutations that it must try to achieve a successful result.

Each vendor's logic cell architecture differs somewhat, but mainly in how support functions such as multiplexing and arithmetic carry terms are implemented. For the most part, engineers do not have to consider the minute details of each logic cell structure, because the conversion of logic into the logic cell is performed by a combination of the HDL synthesis tool and the vendor's proprietary mapping software. In extreme situations, wherein a very specific logic implementation is necessary to squeeze the absolute maximum performance from a specific FPGA, optimizing logic and architecture for a given logic cell structure may have benefits. Engaging in this level of technology-specific optimization, however, can be very tricky and lead to a house-of-cards scenario in which everything is perfectly balanced for a while, and then one new feature is added that upsets the whole plan. If a design appears to be so aggressive as to require fine-tuned optimization, and faster devices cannot be obtained, it may be preferable to modify the architecture to enable more mainstream, abstracted design methodologies.

Notwithstanding the preceding comments, there are high-level feature differences among FPGAs that should be evaluated before choosing a specific device. Of course, it is necessary to pick an FPGA that has sufficient logic and I/O pins to satisfy the needs of the intended application. But not all FPGAs are created equal, despite having similar quantities of logic. While the benefits of one logic structure over another can be debated, the presence or absence of critical design resources can make implementation of a specific design possible or impossible. These resources are clock distribution elements, embedded memory, embedded third-party cores, and multifunction I/O cells.

Clock distribution across a synchronous system must be done with minimal skew to achieve acceptable timing. Each logic cell within a FPGA holds a flop that requires a clock. Therefore, an FPGA must provide at least one global clock signal distributed to each logic cell with low skew across the entire device. One clock is insufficient for most large digital systems because of the proliferation of different interfaces, microprocessors, and peripherals. Typical FPGAs provide anywhere from 4 to 16 global clocks with associated low-skew distribution resources. Most FPGAs do allow clocks to be routed using the general routing resources that normally carry logic signals. However, these paths are usually unable to achieve the low skew characteristics of the dedicated clock distribution network and, consequently, do not enable high clock speeds.

Some FPGAs support a large number of clocks, but with the restriction that not all clocks can be used simultaneously in the same portion of the chip. This type of restriction reduces the complexity of clock distribution on the part of the FPGA vendor because, while the entire chip supports a large number of clocks in total, individual sections of the chip support a smaller number. For example, an FPGA might support 16 global clocks with the restriction that any one quadrant can support only 8 clocks. This means that there are 16 clocks available, and each quadrant can select half of them for arbitrary use. Instead of providing 16 clocks to each logic cell, only 8 need be connected, thus simplifying the FPGA circuitry.

Most FPGAs provide *phase locked loops* (PLLs) or *delay locked loops* (DLLs) that enable the intentional skewing, division, and multiplication of incoming clock signals. PLLs are partially analog circuits, whereas DLLs are fully digital circuits. They have significant overlap in the functions that they can provide in an FPGA, although engineers may debate the merits of one versus the other. The fundamental advantage of a PLL or DLL within an FPGA is its ability to improve I/O timing (e.g., t_{CO}) by effectively removing the propagation delay between the clock input pin and the signal output pin, also known as *deskewing*. As shown in Fig. 11.8, the PLL or DLL aligns the incoming clock to a feedback clock with the same delay as observed at the I/O flops. In doing so, it shifts the incoming