There are two common GAL devices, the 16V8 and the 22V10, although other variants exist as well. They contain eight and ten macrocells each, respectively. The 16V8 provides up to 10 dedicated inputs that feed the AND array, whereas the 22V10 provides 12 dedicated inputs. One of the 22V10's dedicated inputs also serves as a global clock for any flops that are enabled in the macrocells. Output enable logic in a 22V10 is evaluated independently for each macrocell via a dedicated AND term. The 16V8 is somewhat less flexible, because it cannot arbitrarily feed back all macrocell outputs depending on the device configuration. Additionally, when configured for registered mode where macrocell flops are usable, two dedicated input pins are lost to clock and output enable functions.

GALs are fairly low-density PLDs by modern standards, but their advantages of low cost and high speed are derived from their small size. Implementing logic in a GAL follows several basic steps. First, the logic is represented in either graphical schematic diagram or textual (HDL) form. This representation is converted into a netlist using a translation or synthesis tool. Finally, the netlist is fitted into the target device by mapping individual gate functions into the programmable AND array. Given the fixed AND/OR structure of a GAL, fitting software is designed to perform logic optimizations and translations to convert arbitrary Boolean expressions into sum-of-product expressions. The result of the fitting process is a programming image, also called a *fuse map*, that defines exactly which connections, or fuses, are to be programmed and which are to be left at their default state. The programming image also contains other information such as macrocell configuration and other device-specific attributes.

Modern PLD development software allows the back-end GAL synthesis and fitting process to proceed without manual intervention in most cases. The straightforward logic flow through the programmable AND array reduces the permutations of how a given Boolean expression can be implemented and results in very predictable logic fitting. An input signal propagates through the pin and pad structure directly into the AND array, passes through just two gates, and can then either feed a macrocell flop or drive directly out through an I/O pin. Logic elements within a GAL are close to each other as a result of the GAL's small size, which contributes to low internal propagation delays. These characteristics enable the GAL architecture to deliver very fast timing specifications, because signals follow deterministic paths with low propagation delays.

GALs are a logic implementation technology with very predictable capabilities. If the desired logic cannot fit within the GAL, there may not be much that can be done without optimizing the algorithm or partitioning the design across multiple devices. If the logic fits but does not meet timing, the logic must be optimized, or a faster device must be found. Because of the GAL's basic fitting process and architecture, there isn't the same opportunity of tweaking the device as can be done with more complex PLDs. This should not be construed as a lack of flexibility on the part of the GAL. Rather, the GAL does what it says it does, and it is up to the engineer to properly apply the technology to solve the problem at hand. It is the simplicity of the GAL architecture that is its greatest strength.

Lattice Semiconductor's GAL22LV10D-4 device features a worst-case input-to-output combinatorial propagation delay of just 4 ns.* This timing makes the part suitable for address decoding on fast microprocessor interfaces. The same 22V10 part features a 3-ns t_{CO} and up to 250-MHz operation. The t_{CO} specification is a pin-to-pin metric that includes the propagation delays of the clock through the input pin and the output signal through the output pin. Internally, the actual flop itself exhibits a faster t_{CO} that becomes relevant for internal logic feedback paths. Maximum clock frequency specifications are an interesting aspect of all PLDs and some consideration. These specifications are best-case numbers usually obtained with minimal logic configurations. They may define

* GAL22LV10D, 22LV10_04, Lattice Semiconductor, 2000, p. 7.

the highest toggle rate of the device's flops, but synchronous timing analysis dictates that there is more to f_{MAX} than the flop's t_{SU} and t_{CO}. Propagation delay of logic and connectivity between flops is of prime concern. The GAL architecture's deterministic and fast logic feedback paths reduces the added penalty of internal propagation delays. Lattice's GAL22LV10D features an internal clock-to-feedback delay of 2.5 ns, which is the combination of the actual flop's t_{CO} plus the propagation delay of the signal back through the AND/OR array. This feedback delay, when combined with the flop's 3-ns t_{SU}, yields a practical f_{MAX} of 182 MHz when dealing with most normal synchronous logic that contains feedback paths (e.g., a state machine).

11.3 CPLDS

Complex PLDs, or CPLDs, are the mainstream macrocell-based PLDs in the industry today, providing logic densities and capabilities well beyond those of a GAL device. GALs are flexible for their size because of the large programmable AND matrix that defines logical connections between inputs and outputs. However, this anything-to-anything matrix makes the architecture costly to scale to higher logic densities. For each macrocell that is added, both matrix dimensions grow as well. Therefore, the AND matrix increases in a square function of the I/O terms and macrocells in the device. CPLD vendors seek to provide a more linear scaling of connectivity resources to macrocells by implementing a segmented architecture with multiple fixed-size GAL-style logic blocks that are interconnected via a central switch matrix as shown in Fig. 11.5. Like a GAL, CPLDs are typically manufactured with EEPROM configuration storage, making their function nonvolatile. After programming, a CPLD will retain its configuration and be ready for operation when power is applied to the system.

Each individual logic block is similar to a GAL and contains its own programmable AND/OR array and macrocells. This approach is scalable, because the programmable AND/OR arrays remain fixed in size and small enough to fabricate economically. As more macrocells are integrated onto the same chip, more logic blocks are placed onto the chip instead of increasing the size of individual logic blocks and bloating the AND/OR arrays. CPLDs of this type are manufactured by companies including Altera, Cypress, Lattice, and Xilinx.

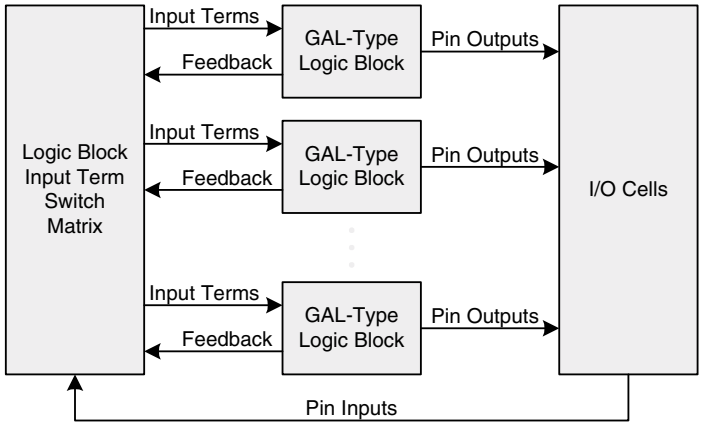

FIGURE 11.5 Typical CPLD architecture.