FPGA PLACEMENT

One thing that stands out in this book's contents: While most individual steps in the compilation flow are covered in a single chapter, placement is covered in three—Chapters 14 through 16. Placement is actually just the problem of assigning specific logic computations to individual logic blocks in the architecture, so why does it merit a longer treatment than, say, FPGA routing? There are at least two reasons.

One reason is historical: Until relatively recently, the placement problem was small enough that structured approaches were possible. These included hand placement, which produced higher-quality results than automatic placement. In contrast, for a problem such as routing, FPGA routers were very fast and efficient, and thus hand-routing was almost never done.

A second reason is that fundamentally different approaches can be taken to solve the placement problem. Do we view the design as an unstructured pile of gates to be scattered across the FPGA's surface, or is there an inherent structure that can be leveraged? And, if we use the computation's structure to drive the placement process, how do we handle portions of the computation, such as control, that likely do not have such an easily determined structure?

These considerations have given rise to several ways of performing FPGA placement, which are represented by the three chapters that follow. In Chapter 14 we consider general-purpose FPGA placement. Such systems, using complex optimization techniques, treat the designer's circuit as essentially an unstructured collection of gates. These are packed together into logic blocks and placed in the array, guided almost exclusively by the design's local connectivity information. Higher-level information, such as the design hierarchy or the regularity in multibit operations, is largely ignored. Thus, these techniques can handle any possible placement problem. Moreover, they serve as a good starting point, as other approaches that rely on more structure in the netlist generally do not work for unstructured designs, and so there must always be some way for unstructured netlists to be processed.

Chapter 15 considers datapath placement. Most designs for an FPGA consist of a large, highly structured datapath and a small, unstructured control system. The datapath is built from multibit function units, such as adders and multipliers, where the computation is fairly similar for each bit of the operands. Datapath-oriented placers can automatically leverage this information to improve the resulting placement quality.

An alternative to fully automatic placement, whether for random logic or for datapaths, is to provide ways for the user to guide the placement process. For example, the user generally knows what portions of the design should be kept

together, where the critical paths are, and how these critical paths should be laid out. Chapter 16 considers such systems, in which placement is more a user-guided process than a fully automated algorithm. Whereas the size of modern FPGA designs, and the increasing quality of placers, is making this approach less attractive over time, constructive placement of critical subsystems is still a valid alternative.