PART III

MAPPING DESIGNS TO RECONFIGURABLE PLATFORMS

The chapters that follow cover the key mapping steps unique to field-programmable gate arrays (FPGAs) and reconfigurable targets. These steps include technology mapping to the primitive FPGA programmable gates (Chapter 13), placement of these gates (Chapters 14 through 16), routing of the interconnect between gates (Chapter 17), retiming of registers in the design (Chapter 18), and bitstream generation (Chapter 19). A final chapter summarizes a number of approaches to accelerating various stages of the mapping process (Chapter 20).

Placement is a difficult mapping problem, but is critical to the performance of the resulting reconfigurable design. As a result, it can be very slow, limiting the rate of the edit–compile–debug loop for reconfigurable application development, and the designs it produces may have longer cycle times than we would like. For these reasons, in addition to the general-purpose algorithms for placement covered in Chapter 14, algorithms that are highly optimized to exploit the regularity of datapaths are discussed in Chapter 15, and constructive approaches to layout are treated in Chapter 16. These more specialized approaches can significantly reduce placement runtime and often deliver placements that allow faster design operation.

As Chapters 13 through 20 demonstrate, there is a well-developed set of approaches and tools for programming reconfigurable applications. However, the tools are always slower than we might like them to be, especially as FPGA capacities continue to grow with Moore's Law. Moreover, the designs they produce are often too large or too slow, and the level at which we must program them is often lower than optimal. These deficiencies present ample opportunities for innovation and improvement in software support for reconfigurable systems.

For the designer who works on reconfiguration issues, the following chapters provide a look under the covers at the tools used to map designs and at the problems they must solve. It is important to understand which problems the tools are and are not solving and how well they can be expected to work. An understanding of the mapping flow and algorithms often helps the designer appreciate why tools may not produce

the quality of results expected and how the design could be optimized to obtain better results. Similarly, understanding the problems that the tools are solving helps the designer understand the trade-offs associated with higher- or lower-level designs and how to mix and match design levels to obtain the desired quality of results with minimal effort.

For the tool or software developer, this part covers the key steps in a traditional tool flow and summarizes the key algorithms used to map reconfigurable designs. With this knowledge the developer can rapidly assimilate conventional approaches and options and thus prepare to explore opportunities to improve quality of results, reduce tool time, or increase automation and raise the configurable design's level of abstraction.