In this chapter we describe how the routing portion of VPR works. We begin by describing the spectrum of FPGA architectures that the router has targeted, and the understandable architecture parameters used to describe an FPGA to VPR. We then explain how a routing architecture is represented internally, and how the succinct description provided by a user is automatically turned into this highly detailed architecture representation. Next, we describe the two routers built into VPR; one is purely routability-driven, while the other is both timing- and routability-driven. The timing-driven router requires a fast and accurate net delay extractor and a path-based timing analyzer, both of which are also discussed. Finally, we compare the performance of VPR to that of several other published CAD tools, and show that it outperforms all the tools to which we have been able to compare.

4.1 Position within the CAD flow

Figure 4.1 shows where the VPR router fits into the CAD flow. Its input is a netlist of logic blocks and an architecture file which describes the target FPGA. A placement is either read in or the VPR placer is used to place the circuit, and then either the routability-driven or the timing-driven router built into VPR is invoked. Each of these routers can perform either a combined global-detailed routing, or global routing only. Once routing is complete, various statistics, such as routed wirelength, routing resource utilization and the delay of the critical path are output.
CHAPTER 4 Routing Tools and Routing Architecture Generation

Placement and routing output files, Placement and routing statistics

FIGURE 4.1 VPR router operation and position within the CAD flow.

4.2 Architecture Parameterization and Generation

Since our primary goal in this research is to investigate many different FPGA architectures, we would like our tools to:

1. Be as architecture-independent as possible, and
2. Be easy to use with different architectures — that is, we would like to be able to describe different architectures to VPR quickly.

To achieve the first goal, we have made the router, graphics routines, delay extractor, and various other routines all operate on a directed graph that describes the FPGA. This routing-resource graph representation is very general, and can describe a wide variety of FPGA architectures. Unfortunately, describing a new architecture by creating a routing-resource graph by hand is not feasible — the routing-resource graph to describe a typical FPGA containing 8000 4-LUTs is almost 30 MB in size. One possibility is to design a basic tile (a single logic block and its associated routing) manually, and create a program to automatically replicate and stitch together this tile into a routing-resource graph describing the entire FPGA. Even creating a basic tile manually is too time-consuming for our purposes, however. A typical tile contains several hundred programmable switches and wires, so it can take hours or days to describe even one tile. Furthermore, such a hand-crafted tile is designed for one value of routing channel width, W. In many of our experiments we wish to vary W in order to see