5 A RETARGETABLE TIMING ANALYSIS TOOL — CINDERELLA

5.1 INTRODUCTION

We have implemented the timing analysis technique described in the previous two chapters in a tool called cinderella. Cinderella automatically formulates the ILP problem and passes it to the ILP solver. With cinderella, we can analyze significantly large programs. Initial results from cinderella helped us to fine-tune the formulation of cache constraints and to experiment with different ways of providing the functionality constraints. These experiments improve both the solving time and accuracy of the timing analysis. Our goal is to make the tool robust, extensible, and easy to use so that external researchers can make use of it. In this chapter, we will discuss the issues encountered in implementing cinderella, with emphasis on the retargetability of the tool. The software architecture of cinderella will be discussed, and an example showing the run-time environment of the tool will be given. This chapter will be helpful for those who plan to use the tool or to port the tool to model other microprocessor architectures.

In embedded system designs, it is common to evaluate the software performance running on a number of different processors, memory systems, and peripherals so as to select a cost-effective system satisfying all real-time constraints. Thus, for practical
uses, the timing analysis tool must be easily retargetable to model programs running on different hardware systems. The timing analysis technique described in the previous chapters models common microarchitecture features that are present in many popular processors. It is not tied to any specific processor platform. By implementing cinderella in a modular way, the tool can be easily retargeted to model a wide range of systems.

In the following sections, we will first state the minimum set of information needed for the timing analysis. Next, we will discuss where and how to extract this information. Interestingly, the needed information can be obtained in quite a few ways, each with different trade-offs among the retargetability of the tool, the implementation effort, and the program code quality. These issues have thus far not been thoroughly studied. We will compare the techniques used in extracting the information and explain the rationale for the one used in cinderella.

Once we have decided how to extract the information, the next important step is to establish a robust interface between the target dependent modules which extracts the information, and the target independent core which processes the information to generate the ILP problem for solving. This interface is crucial to cinderella. On one hand, it has to hide any target specific information from the core so that the core can work with any target system. On the other hand, it must allow the target dependent module to supply enough information to the core for accurate analysis. Furthermore, the interface should be designed in such a way to minimize the programming effort needed to retarget the tool. This means that the retargetable modules should be simple and the core should perform as many functions as possible.

As described in Chapter 3, the program path annotations are very important in tightening the estimated bound. However, in order to provide accurate annotations, the user must have in-depth knowledge of the program inputs and program control flow. Cinderella's graphical interface displays the program code both in source and assembly level. This helps the user to gather program control flow information and to examine code changes due to compiler optimizations. Both of these are important in helping the user provide accurate functionality constraints. The user can also view the results and identify the bottlenecks in the program easily.

In the following sections, the issues involved in designing the retargetable back-ends are discussed. Different approaches will be compared. Then in Section 5.5, we will provide an example showing the operations of cinderella.

5.2 ISSUES IN TIMING ANALYSIS

In this section, we will discuss the issues in implementing the timing analysis tool. We will describe and contrast how other researchers deal with these problems. To start with, we need to understand what information is needed for extreme case timing analysis.