9 HARDWARE/SOFTWARE PARTITIONING USING THE LYCOS SYSTEM

Jan Madsen, Jesper Grode, and Peter V. Knudsen

Department of Information Technology
Technical University of Denmark
Lyngby, Denmark

9.1 INTRODUCTION

Hardware/software partitioning is often viewed as the synthesis of a target architecture consisting of a single processor and a single dedicated hardware component (full custom, FPGA, etc.) from an initial system specification, e.g. as in [289]. Even though the **single processor**, single **dedicated hardware** architecture is a special and limited example of a distributed system, the architecture is relevant in many areas such as **DSP** design, construction of embedded systems, software execution acceleration and hardware emulation and prototyping [290], and it is the most commonly used target architecture for automatic hardware/software partitioning.

J. Staunstrup and W. Wolf (eds.), Hardware/Software Co-Design: Principles and Practice, 283-305.

In this chapter we present the LYCOS (LYngby CO-Synthesis) system which is an experimental hardware/software co-synthesis system. In its current version, LYCOS may be used for hardware/software partitioning using a target architecture as described above. In this chapter we will focus on how LYCOS is used to do design space exploration in codesign. Details about the algorithms used within LYCOS can be found elsewhere [291, 292, 293, 294].

The chapter is organized as follows. Section 9.2 presents the problem of hardware/software partitioning and its relation to design space exploration. In Section 9.3 we give an overview of the LYCOSsystem. Section 9.4 gives a step by step walk-through of a partitioning session in LYCOS. Section 9.5 describes how LYCOSmay be used for design space exploration. Finally, Section 9.6 provides an evaluation of the system and some conclusions on the current version of LYCOS.

9.2 PARTITIONING AND DESIGN SPACE EXPLORATION

Consider the hardware/software **partitioning** and a set of requirements to be fulfilled by the partitioned system. In order to obtain a feasible partition, that is, a partition which fulfills the requirements, we need to know the target architecture, i.e. the processor on which to run the software, the technology of the dedicated hardware, and the interface used for communication between hardware and software. However, the choice of target architecture will greatly influence the outcome of the partition as each target architecture will have different "best" partitions. For instance, selecting a *fast* but expensive processor may lead to little (or no) hardware needed, while selecting a *slow* but cheap processor may require a large amount of dedicated hardware. Thus, in order to solve the problem efficiently we need a way to explore the design space.

Typically we will have an idea about possible suitable target architectures. These may be selected based on the designer's experience, the desire to reuse predesigned components or the use of third party components. One way to explore the design space is to find the best partition for each possible target architecture and select the best among these.

To find the best partition for a given target architecture, we need a model to represent computation and ways to estimate metrics of software, hardware, and communication.

It is important that the model of computation is independent of any particular implementation strategy, that being software or hardware. This will allow for an unbiased design space exploration as well as for translations from various specification languages. One model targeted towards partitioning is based on extracting chunks of computation, called basic scheduling blocks or just blocks. A partition in this model is an enumeration of each block indicating whether