Chapter 6
Trends in Reconfigurable Embedded Systems

C. Valderrama
University of Mons, Belgium

L. Jojczyk
University of Mons, Belgium

P. Possa
University of Mons, Belgium

ABSTRACT
This chapter presents reconfigurable embedded systems by looking closely at three different but interrelated aspects: design tools, methodologies and architectures, paying special attention at reconfigurable interconnections. The authors will start having a closer look at the evolution of the latest design strategies, tools and methodologies facing challenges and user requirements. Reconfigurable interconnections will be analyzed, examining topologies, drawbacks, and capabilities, specially focusing on the reconfiguration potential. From the application point of view, the authors will resume with a case study regarding embedded systems, a Software-Defined Radio application highlighting the most significant technical features and design choices.

INTRODUCTION
Embedded systems are designed to perform dedicated specific tasks with real-time processing constraints. Such systems comprise complete devices ranging from portable such as video cameras and set-top boxes, to complex industrial controllers including mechanical parts.

Embedded systems complexity varies from a single microcontroller to several interrelated dedicated functional units, peripherals and network devices. Often they come with contradictory design constraints such as low power consumption and high performance. Indeed, mass production portable devices should be cheap and power efficient. On the other hand, they still need to support multiple, sometimes real-time, applications and
standards. The wide spectrum of design requirements leads to complex architectures composed by several processors and customized hardware components. These multiprocessors System-on-chip (MSoC) have now become the foundation of most common appliances such as digital TV, navigation systems and wireless devices.

One of the major challenges on embedded systems design is to obtain the necessary system performance with efficient utilization of hardware resources. To accomplish this goal, the system must be optimized in power consumption, size and cost, complying with flexibility, reliability and processing performance requirements. The architecture must support multiple applications and standards. Targeting the architecture to a single application prevents its evolution and flexibility. Accuracy at the early design stage is not an option; low level simulation suffers from low simulation speed and design effort. Such conflicting design requirements cannot be satisfied by traditional design methods (Pimentel, Lieverse, & Van der Wolf, 2001).

Classical hardware/software codesign methodology was typically based on system level specification refinement forcing the designer to make early decisions regarding system architecture (Coumeri & Thomas, 1995) (Kim, et al., 1995) (Bauer & Ecker, 1997) (Balarin, et al., 1997) (Hines & Borriello, 1997). A system-level model represents application behavior, architecture characteristics and the relation between application and architecture (application partitioning, mapping and resources utilization). One essential design task is to perform design space exploration at the early design stages. System level simulation is here used to provide a first estimate on system performance (for instance, power consumption or cost) and to make early decisions on various design parameters (Pimentel, Erbas, & Polstra, 2006) (Balarin, Watanabe, Hsieh, Lavagno, Passerone, & Sangiovanni-Vincentelli, 2003) (Stammermann, Kruse, Nebel, & Pratsch, Oct. 2001) (Gadient, Debardelaben, & Madisetti, 1997). Using a high abstraction level minimizes the modeling effort increasing simulation speed. Several possible implementations can be rapidly evaluated at the system-level prior to the refinement (at the instruction-level, cycle-accurate or register-transfer level) and synthesis.

Recent work, Y-chart methodology, recognizes a clear separation between application and architecture models: an explicit mapping step relates the application model to the architecture model (Balarin, et al., 1997) (Kienhuis, Depretere, Vissers, & Van der Wolf, 1997) (Lieverse, Stefanov, Van der Wolf, & Deprettere, Nov. 2001) (Zivkovic, Van der Wolf, Deprettere, & De Kock, Oct. 2002) (Mohanty & Prasanna, 2002) (Balarin, Watanabe, Hsieh, Lavagno, Passerone, & Sangiovanni-Vincentelli, 2003). The application model describes only functional behavior of an application, independently of architecture characteristics or timing parameters. The application concerns are clearly separated in computational and communication events. The architecture model defines architecture resources, capturing timing characteristics and performance consequences of an application event. The latter involves simulation of programmable as well as reconfigurable or dedicated components. The mapping of each application onto an architecture instance is evaluated by a performance analysis. The resulting performance numbers may inspire the designer to improve the architecture, restructure the application, or change the mapping.

Design space exploration turned into the platform-based design (PBD) approach at the architecture level (Keutzer, Malik, Newton, Rabaey, & Sangiovanni-Vincentelli, 2000) (Carloni, De Bernardinis, Pinello, Sangiovanni-Vincentelli, & Sgroi, 2005) (Cooke, McNelly, & Martin, 1999). The platform is an abstraction layer hiding the details of several possible implementation scenarios. The approach makes use of a common platform and intellectual property (IP) blocks shared among multiple applications. An IP block is here characterized by a model representing its
Related Content

Eliciting Data Warehouse Contents for Policy Enforcement Rules
[www.igi-global.com/article/eliciting-data-warehouse-contents-for-policy-enforcement-rules/112041?camid=4v1a](www.igi-global.com/article/eliciting-data-warehouse-contents-for-policy-enforcement-rules/112041?camid=4v1a)

Modeling Business and Requirements Relationships to Facilitate the Identification of Architecturally Significant Requirements
[www.igi-global.com/article/modeling-business-and-requirements-relationships-to-facilitate-the-identification-of-architecturally-significant-requirements/111447?camid=4v1a](www.igi-global.com/article/modeling-business-and-requirements-relationships-to-facilitate-the-identification-of-architecturally-significant-requirements/111447?camid=4v1a)

Platform Independent Analysis of Probabilities on Multithreaded Programs
[www.igi-global.com/article/platform-independent-analysis-of-probabilities-on-multithreaded-programs/103281?camid=4v1a](www.igi-global.com/article/platform-independent-analysis-of-probabilities-on-multithreaded-programs/103281?camid=4v1a)

Mitigating Type Confusion on Java Card
[www.igi-global.com/article/mitigating-type-confusion-java-card/77915?camid=4v1a](www.igi-global.com/article/mitigating-type-confusion-java-card/77915?camid=4v1a)