Preface

Nowadays, the prevalence of electronic and computing systems in our lives is so ubiquitous that it would not be far-fetched to state that we live in a cyber-physical world dominated by computer systems. Examples include pacemakers implanted within the human body to regulate and monitor heartbeats, cars and airplanes transporting us, smart grids, and traffic management.

All these systems demand more and more computational performance to process large amounts of data from multiple data sources, and some of them with guaranteed processing response times; in other words, systems required to deliver their results within pre-defined (and sometimes extremely short) time bounds. This timing aspect is vital for systems like planes, cars, business monitoring, e-trading, etc. Examples can be found in intelligent transportation systems for fuel consumption reduction in cities or railways, or autonomous driving of vehicles. All these systems require processing and actuation based on large amounts of data coming from real-time sensor information.

As a result, the computer electronic devices which these systems depend on are constantly required to become more and more powerful and reliable, while remaining affordable. In order to cope with such performance requirements, chip designers have recently started producing chips containing multiple processing units, the so-called multi-core processors, effectively integrating multiple computers within a single chip, and more recently the many-core processors, with dozens or hundreds of cores, interconnected with complex networks on chip. This radical shift in the chip design paved the way for parallel computing: rather than processing the data sequentially, the cooperation of multiple processing elements within the same chip allows systems to be executed concurrently, in parallel.

Unfortunately, the parallelization of the computing activities brought up many challenges, because it affects the timing behavior of the systems as well as the entire way people think and design computers: from the design of the hardware architecture, through the operating system up to the conceptualization of the end-user application. Therefore, although many-core processors are promising candidates to improve the responsiveness of these systems,

the interactions that the different computing elements may have within the chip can seriously affect the performance opportunities brought by parallel execution. Moreover, providing timing guarantees becomes harder, because the timing behavior of the system running within a many-core processor depends on interactions that are most of the time not known by the system designer. This makes system analysts struggle in trying to provide timing guarantees for such platforms. Finally, most of the optimization mechanisms buried deep inside the chip are geared only to increase performance and execution speed rather than providing predictable time behavior.

These challenges need to be addressed by introducing predictability in the vertical stack from high-level programming models to operating systems, together with new offline analysis techniques. This book covers the main techniques to enable performance and predictability of embedded applications. The book starts with an overview of some of the current many-core embedded platforms, and then addresses how to support predictability and performance in different aspects of computation: a predictable parallel programming model, the mapping and scheduling of real-time parallel computation, the timing analysis of parallel code, as well as the techniques to support predictability in parallel runtimes and operating systems.

The work reflected in this book was done in the scope of the European project P-SOCRATES, funded under the FP7 framework program of the European Commission. The project website (www.p-socrates.eu), provides further detailed information on the techniques presented here. Moreover, a reference implementation of the methodologies and tools was released as the UpScale Software Development Kit (http://www.upscale-sdk.com).

Luís Miguel Pinho
Eduardo Quiñones
Marko Bertogna
Andrea Marongiu
Vincent Nélis
Paolo Gai
Juan Sancho

February 2018