Preface

Preface and Outline

Computing systems are around for a relatively short period; it is only since the invention of the microprocessor systems in the early seventies that processors became affordable by everybody. Although this period is short it is hardly imaginable how current life would be without computing systems. They penetrated virtually all aspects of life. Most visible are PCs, smartphones and gaming consoles. Almost every household has several of them, counting up to billions worldwide. However, we are even more dependent on so-called embedded systems; that are computing systems which are usually not visible, but they largely determine the functionality of the surrounding system or equipment. They control factories, take care of security (using e.g. smart cameras), control your car and its engine (actually your car likely contains tens of embedded systems), calculate your travel route, take care of internet traffic, control your coffee machine, etc.; they make the surrounding system behave more intelligently. As a consequence we are dependent on them.

One of the reasons computing systems are everywhere is that they are cheap, and they are cheap because of the ongoing integration and miniaturization. While we needed a large room to install the earliest computing systems, now a processor can be integrated in a few square millimeters or less, while giving substantial performance in terms of operations per second.

Talking about performance, performance demands are constantly increasing. This holds especially for multi-media type of systems, that are systems which process streams of data, data being e.g. audio, all kinds of sensing data, video and graphics. To give an example, the next generation of smartphones is estimated to require about one Tera operations per second for its video, vision, graphics, GPS, audio and speech recognition capabilities, and to make matters worse, this performance has be delivered in a small package and should leave your batteries operational for many days. To deliver this performance for such a small energy budget, computing systems will contain many processors. Some of these processors will be general purpose programmable; others need to be more tuned to the application domain in order to reach the required computational efficiency (in operations

per second per Watt). Future multi-media systems will therefore be heterogeneous multi-core.

Multi-media will not be the sole domain of advanced smartphones and the like. Since cameras are getting very cheap we predict that many future smart systems will be enhanced by adding vision capabilities like surveillance, recognition, virtual reality, visual control, and so on. Multi-media systems can be dedicated for one specific application; however an ongoing trend is to map multiple applications to the same system. This sharing of resources by several applications makes the system cheaper and more versatile, but substantially adds to its design complexity.

Although integration is still following Moore's law, making computing systems cheaper, at least when counting chip area, their design becomes extremely complex, and therefore very costly. This is not only caused by increased functionality and performance demands of the running applications, but also by other, non-functional, requirements like energy cost and real-time demands. Making systems functionally correct is already quite challenging, but making this functionality operating at the right speed, obeying severe throughput and latency requirements can become a nightmare and may lead to many and large debugging and redesign cycles. Therefore a more systematic design method is urgently needed which avoids large debugging cycles while taking real-time demands into account as an integral part of the design process.

Aim of This Book

In this book we focus on (streaming) multi-media systems, and in particular on the real-time aspects of these systems. These systems run multiple applications and are realized using multiple processing cores, some of them specialized for a specific target application domain. The goal of this book is to make you familiar with techniques to model and analyze these systems, both hardware and software, while taking timing requirements into account. The models will be taken from the data flow modeling domain; in particular we will teach you how to use SDF (Synchronous Data Flow) models to specify and analyze your system. Although SDF is restricted in its expressive power, it has very good analysis power, and appropriate tools exist to perform this analysis. For example, timing properties and deadlock avoidance can be easily verified. It also allows for the calculation of appropriate buffer sizes for the inter-core communication buffers. Based on this specification and analysis this book teaches you how to synthesize and implement the specified multi-processor system using FPGA technology. This synthesis is correct by construction, and therefore it avoids many debugging iterations. The FPGA implementation can also act as a quick prototype for a final silicon realization.

Mapping multiple applications to such a system requires a run-time manager. This manager is responsible for admission control, i.e., can a new application be added to the other, already running applications, such that every application still

meets its timing requirements. Once admitted the run-time manager is also responsible for controlling the resources and enforcing the right time budgets to all applications. This book will show several techniques for performing these management tasks.

You may not be satisfied by just running one set of applications. The set of running applications may regularly change, leading to multiple use cases. This adds new dimensions to the design process, especially when mapping to FPGAs; some of them will be treated at the end of this book. E.g. how do you share the resources of an FPGA between the multiple use cases, this to reduce the number of FGPA reconfigurations. Another dimension discussed is the estimation of the amount of FPGA resources needed by a set of use cases prior to the synthesis.

In short you will learn how to map multiple applications, possibly divided into multiple use cases to a multi-processor system, and you will be able to quickly realize such a system into an FPGA. All the theory discussed in this book is supported by a complete design flow called MAMPS, which stands for Multiple Applications Multi-Processor Synthesis. The flow is fully implemented and demonstrated by several examples.

Within the book we make several restrictions. A major one is that we mainly deal with soft real-time constraints. The techniques used in this book do not give hard real-time guarantees (unless indicated otherwise). The reason is that giving hard real-time guarantees may result in severe overestimation of the required resources, and therefore may give a huge performance and cost penalty. This does not mean that we do not emphasize research in hard real-time systems. On the contrary; it is one of our major research themes, already for many years, and the focus of many of our current projects. The reader is referred to the book website (see below) for further information on our other projects.

Audience

This book covers a complete design trajectory for the design of multi-media systems running multiple applications; it includes both theory and practice. It is meant for all people interested in designing multi-media and other real-time multi-processor systems. It helps them to think and reason about timing requirements and offers them various modeling, analysis, design and management techniques needed to realize these complex systems first time right. The book is also meant for system level architects who want to quickly make high level estimates and system trade-offs based on solid modeling. The book is also suitable for use within a post graduate course. To this purpose we included extensive introductory chapters on trends and challenges in multi-media systems, and on the theory behind application modeling and scheduling. In particular data flow techniques are treated in some depth. In such a course the available tools will help students to get familiar with future design flows, and bring the theory into practice.

Accompanying Material

Besides the printed copy, there is an accompanying book website at http://www.es.ele.tue.nl/~akash/MMSBook. This website contains further information about designing real-time systems and various links to other related research. In addition to that, accompanying slides can be found on the website. The slides can be used in a course, provided the copyright information is retained.

As mentioned most of the presented techniques and algorithms are integrated in the MAMPS design flow. The corresponding tooling, and its documentation, can be found at http://www.es.ele.tue.nl/MAMPS. The tools can be used online. The site contains a couple of tested examples to try out the tools. For collaborating partners tools can be made available on request for development.

This book is based on research and development being performed at the TU/e, the Eindhoven University of Technology, within the PreMaDoNA project of the Electronic Systems group. PreMaDoNA stands for predictable matching of demands on networked architectures. Other results from this project and it's follow up projects can also be found following the links on the book website.

Organization of This Book

This book is divided into seven chapters. The first two are introductory. The first one describes trends in multimedia systems; the second one goes into the theory behind data flow modeling and scheduling, and introduces the necessary notation. Chapter 3 describes our new iterative analysis method. Chapter 4 treats how to perform resource management. Chapter 5 describes the MAMPS design flow which allows for quick realization of a system into an FPGA. Chapter 6 extends the system to support multiple use cases. Finally Chap. 7 gives several conclusions and outlines the open problems that are not solved in this book. Although the best way is to read all chapters in the presented order, some readers may find it convenient to skip parts on first reading. Chapters 3, 4 and 5 do not have big interdependences. Therefore, after reading the first 2 chapters, readers can select continue with either Chap. 3, 4 or 5. Chapter 6 depends on Chap. 5, so readers interested in mapping multiple use cases should first read Chap. 5.

Henk Corporaal

http://www.springer.com/978-94-007-0082-6

Multimedia Multiprocessor Systems
Analysis, Design and Management
Kumar, A.; Corporaal, H.; Mesman, B.; Ha, Y.
2010, XVI, 164 p., Hardcover
ISBN: 978-94-007-0082-6