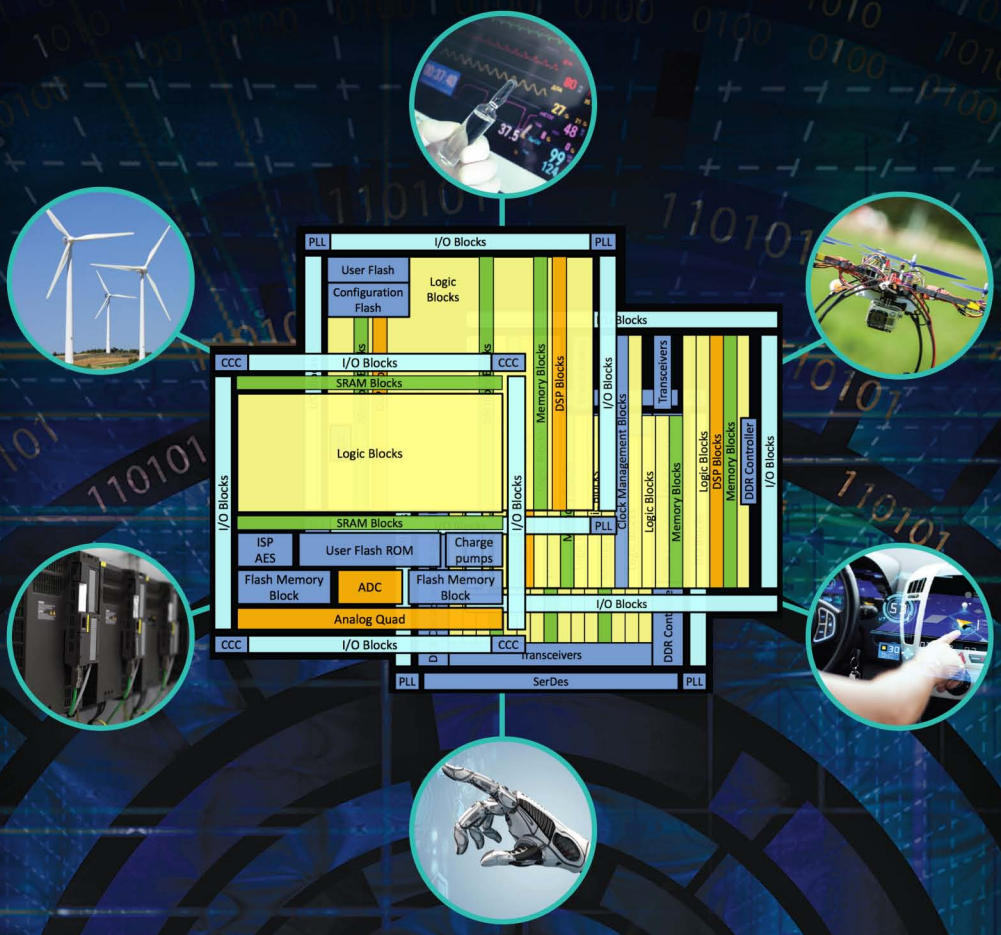


# FPGAs

## Fundamentals, Advanced Features, and Applications in Industrial Electronics

Juan José Rodríguez Andina • Eduardo de la Torre Arnanz
María Dolores Valdés Peña

CRC Press
Taylor & Francis Group

# FPGAs

## Fundamentals, Advanced Features, and Applications in Industrial Electronics

# FPGAs
## Fundamentals, Advanced Features, and Applications in Industrial Electronics

Juan José Rodríguez Andina,
Eduardo de la Torre Arnanz, and
María Dolores Valdés Peña

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742



Printed on acid-free paper
Version Date: 20161025

International Standard Book Number-13: 978-1-4398-9699-0 (Hardback)

# Contents

# *Preface*

This book intends to contribute to a wider use of field-programmable gate arrays (FPGAs) in industry by presenting the concepts associated with this technology in a way accessible for nonspecialists in hardware design so that they can analyze if and when these devices are the best (or at least a possible) solution to efficiently address the needs of their target industrial applications. This is not a trivial issue because of the many different (but related) factors involved in the selection of the most suitable hardware platform to solve a specific digital design problem. The possibilities enabled by current FPGA devices are highlighted, with particular emphasis on the combination of traditional FPGA architectures and powerful embedded processors, resulting in the so-called field-programmable systems-on-chip (FPSoCs) or systems-on-programmable-chip (SoPCs). Discussions and analyses are focused on the context of embedded systems, but they are also valid and can be easily extrapolated to other areas.

The book is structured into nine chapters:

- Chapter 1 analyzes the different existing design approaches for embedded systems, putting FPGA-based design in perspective with its direct competitors in the field. In addition, the basic concept of FPGA "programmability" or "configurability" is discussed, and the main elements of FPGA architectures are introduced.
- From the brief presentation in Chapter 1, Chapter 2 describes in detail the main characteristics, structure, and *generic* hardware resources of modern FPGAs (logic blocks, I/O blocks, and interconnection resources). Some specialized hardware blocks (clock management blocks, memory blocks, hard memory controllers, transceivers, and serial communication interfaces) are also analyzed in this chapter.
- Embedded soft and hard processors are analyzed in Chapter 3, because of their special significance and the design paradigm shift they caused as they transformed FPGAs from hardware accelerators to FPSoC platforms. As shown in this chapter, devices have evolved from simple ones, including one general-purpose microcontroller, to the most recent ones, which integrate several (more than 10 in some cases) complex processor cores operating concurrently, opening the door for the implementation of homogeneous or heterogeneous multicore architectures. The efficient communication between processors and their peripherals is a key factor to successfully develop embedded systems. Because of this, the currently available on-chip buses and their historical evolution are also analyzed in detail in this chapter.

- Chapter 4 analyzes DSP blocks, which are very useful hardware resources in many industrial applications, enabling the efficient implementation of key functional elements, such as digital filters, encoders, decoders, or mathematical transforms. The advantages provided by the inherent parallelism of FPGAs and the ability of most current devices to implement floating-point operations in hardware are also highlighted in this chapter.
- Analog blocks, including embedded ADCs and DACs, are addressed in Chapter 5. They allow the functionality of the (mostly digital) FPGA devices to be extended to simplify interfacing with the analog world, which is a fundamental requirement for many industrial applications.
- The increasing complexity of FPGAs, which is clearly apparent from the analyses in Chapters 2 through 5, can only be efficiently handled with the help of suitable software tools, which allow complex design projects to be completed within reasonably short time frames. Tools and methodologies for FPGA design are presented in Chapter 6, including tools based on the traditional RTL design flow, tools for SoPC design, high-level synthesis tools, and tools targeting multithread accelerators for high-performance computing, as well as debugging and other auxiliary tools.
- There are many current applications where tremendous amounts of data have to be processed. In these cases, communication resources are key elements to obtain systems with the desired (increasingly high) performance. Because of the many functionalities that can be implemented in FPGAs, such efficient communications are required to interact not only with external elements but also with internal blocks to exchange data at the required rates. The issues related to both off-chip and in-chip communications are analyzed in detail in Chapter 7.
- The ability to be reconfigured is a very interesting asset of FPGAs, which resulted in a new paradigm in digital design, allowing the same device to be readily adapted during its operation to provide different hardware functionalities. Chapter 8 focuses on the main concepts related to FPGA reconfigurability, the advantages of using reconfiguration concurrently with normal operation (i.e., at run time), the different reconfiguration alternatives, and some existing practical examples showing high levels of hardware adaptability by means of run-time dynamic and partial reconfiguration.
- Today, FPGAs are used in many industrial applications because of their high speed and flexibility, inherent parallelism, good cost–performance trade-off (offered through wide portfolios of different device families), and the huge variety of available specialized

logic resources. They are expected not only to consolidate their application domains but also to enter new ones. To conclude the book, Chapter 9 addresses industrial applications of FPGAs in three main design areas (advanced control techniques, electronic instrumentation, and digital real-time simulation) and three very significant application domains (mechatronics, robotics, and power systems design).

# *Acknowledgments*

The authors have greatly benefited during their more than 25 years of experience in FPGA design from advice and comments from, and discussions with, many colleagues, from both academia and the industry. Citing all of them individually here is not possible and might result in some unintentional omission. We hope all of them know they are represented here through our grateful acknowledgments to our present and past colleagues and students at the Department of Electronic Technology, University of Vigo, and the Center of Industrial Electronics, Technical University of Madrid; the people at the many companies for which we have consulted and developed projects in the area; our colleagues in the IEEE Industrial Electronics Society; and those we have met over the years in many scientific forums, such as IECON, ISIE, ICIT, FPL, Reconfig, and ReCoSoc.

Last, but of course not the least, our final word of gratitude goes to our families for their unconditional support.

# *Authors*

**Juan José Rodríguez Andina** received his MSc from the Technical University of Madrid, Spain, in 1990, and his PhD from the University of Vigo, Spain, in 1996, both in electrical engineering. He has also received the Extraordinary Doctoral Award from the University of Vigo. He is an associate professor in the Department of Electronic Technology, University of Vigo. In 2010–2011, he was on sabbatical as a visiting professor at the Advanced Diagnosis, Automation, and Control Laboratory, Electrical and Computer Engineering Department, North Carolina State University, Raleigh. He has been working for more than 25 years in digital systems design, with emphasis on FPGA-based design for industrial applications. He has authored more than 140 journal and conference articles and holds several Spanish, European, and U.S. patents. He currently serves as vice president for conference activities of the IEEE Industrial Electronics Society and has been general chair, technical program chair, and member of other various committees in a number of IEEE conferences (such as IECON, ISIE, ICIT, and INDIN), where he regularly organizes special sessions related to industrial applications of FPGAs and embedded systems. He is the former editor-in-chief of the *IEEE Industrial Electronics Magazine* and an associate editor for *IEEE Transactions on Industrial Electronics* and *IEEE Transactions on Industrial Informatics*.

**Eduardo de la Torre Arnanz** is an associate professor of electronics since 2002 and obtained his MSc and PhD in electrical engineering from the Technical University of Madrid in 1989 and 2000, respectively. His main expertise is in FPGA-based design and, in particular, in partial and dynamic reconfiguration of digital systems and reconfigurable hardware acceleration. He has been working for more than 25 years on digital systems design, among which more than 20 have been around FPGAs, mostly in industrial applications. He has authored more than 40 papers on reconfigurable systems in the last five years and has been program cochair of ReCoSoC (2015), Reconfig (2012 and 2013), DASIP (2013), and SPIE VLSI Circuits & Systems (2009 and 2011) conferences as well as a program committee member

of conferences such as FPL, ReCoSoC, RAW, WRC, ISVLSI, and SIES. He is also a reviewer of numerous conferences and journals such as the *IEEE Transactions on Computers*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Industrial Electronics*, and *Sensor* magazine.

**María Dolores Valdés Peña** is an associate professor in the Department of Electronic Technology, University of Vigo, Spain. She received her MSc from Universidad Central de Las Villas, Santa Clara, Cuba, in 1990, and her PhD from the University of Vigo, Vigo, Spain, in 1997, both in electrical engineering. She received the Extraordinary Doctoral Award from the University of Vigo. In 1998, the Society of Instrument and Control Engineers (SICE) of Japan gave her the award for the best research work at the 37th Annual SICE Conference. Over the years, she has authored more than 120 journal and conference articles. Her research interests include the design of reconfigurable systems based on FPGAs applied to data acquisition and conditioning systems, digital signal processing and control, wireless sensor networks, and field-programmable systems-on-chip for industrial applications.

# 1

# *FPGAs and Their Role in the Design of Electronic Systems*

## 1.1 Introduction

This book is mainly intended for those users who have had certain experience in digital control systems design, but for some reason have not had the opportunity or the need to design with modern field-programmable gate arrays (FPGAs). The book aims at providing a description of the possibilities of this technology, the methods and procedures that need to be followed in order to design and implement FPGA-based systems, and selection criteria on what are the best suitable and cost-effective solutions for a given problem or application. The focus of this book is on the context of embedded systems for industrial use, although many concepts and explanations could be also valid for other fields such as high-performance computing (HPC). Even so, the field is so vast that the whole spectrum of specific applications and application domains is still tremendously large: transportation (including automotive, avionics, railways systems, naval industry, and any other transportation systems), manufacturing (control of production plants, automated manufacturing systems, etc.), consumer electronics (from small devices such as an air-conditioning remote controller to more sophisticated smart appliances), some areas within the telecom market, data management (including big data), military industry, and so forth.

In this chapter, the concept of embedded systems is presented from a wide perspective, to later show the ways of approaching the design of embedded systems with different complexities. After introducing all possibilities, the focus is put on FPGA-related applications. Then, the basic concept of FPGA "programmability" or "configurability" is discussed, going into some description of the architectures, methods, and supported tools required to successfully carry out FPGA designs with different complexities (not only in terms of size but also in terms of internal features and design approaches).

## 1.2 Embedded Control Systems: A Wide Concept

Embedded control systems are, from a very general perspective, control elements that, in a somewhat autonomous manner, interact with a physical system in order to have an automated control over it. The term "embedded" refers to the fact that they are placed in or nearby the physical system under control. Generally speaking, the interfaces between the physical and control systems consist of a set of sensors, which provide information from the physical system to the embedded system, and a set of actuators capable, in general, of modifying the behavior of the physical system.

Since most embedded systems are based on digital components, signals obtained from analog sensors must be transformed into equivalent digital magnitudes by means of the corresponding analog-to-digital converters (ADCs). Equivalently, analog actuators are managed from digital-to-analog converters (DACs). In contrast, digital signals do not require such modifications. The success of smart sensor and actuator technologies allows such interfaces to be simplified, providing standardized communication buses as the interface between sensors/actuators and the core of the embedded control system.

Without loss of generality regarding the earlier paragraphs, two particular cases are worth mentioning: communication and human interfaces. Although both would probably fit in the previously listed categories, their purposes and nature are quite specific.

On one hand, communication interfaces allow an embedded system to be connected to other embedded systems or to computing elements, building up larger and more complex systems or infrastructures consisting of smaller interdependent physical subsystems, each one locally controlled by their own embedded subsystem (think, for instance, of a car or a manufacturing plant with lots of separate, but interconnected, subsystems).

Communication interfaces are "natural" interfaces for embedded control systems since, in addition to their standardization, they take advantage from the distributed control system philosophy, providing scalability, modularity, and enhanced dependability—in terms of maintainability, fault tolerance, and availability as defined by Laprie (1985).

On the other hand, human interfaces can be considered either like conventional sensors/actuators (in case they are simple elements such as buttons, switches, or LEDs) or like simplified communication interfaces (in case they are elements such as serial links for connecting portable maintenance terminals or integrated in the global communication infrastructure in order to provide remote access). For instance, remote operation from users can be provided by a TCP/IP socket using either specific or standard protocols (like http for web access), which easily allows remote control to be performed from a web browser or a custom client application, the server being embedded in the control system. Nowadays, nobody gets surprised by the possibility of

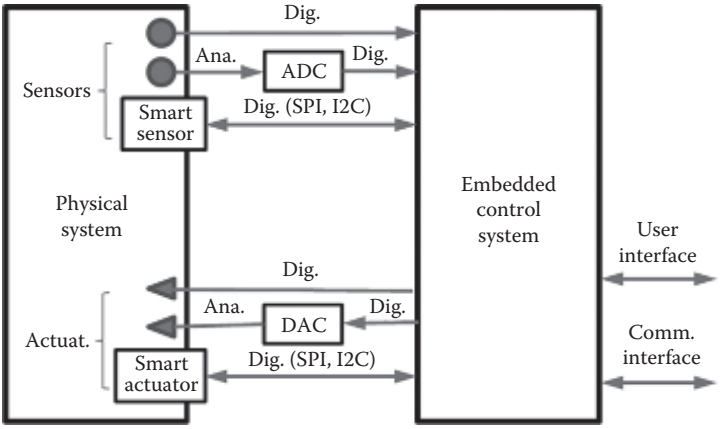


**FIGURE 1.1**
Generic block diagram of an embedded system.

using a web browser to access the control of a printer, a photocopy machine, a home router, or a webcam in a ski resort.

Figure 1.1 presents a general diagram of an embedded control system and its interaction with the physical system under control and other subsystems.

Systems based on analog sensors and actuators require signal conditioning operations, such as low-noise amplification, anti-aliasing filtering, or filtering for noise removal, to be applied to analog signals. Digital signal processing and computationally demanding operations are also usually required in this case. On the other hand, discrete sensors and actuators tend to make the embedded system more control dependent. Since they have to reflect states of the system, complexity in this case comes from the management of all state changes for all external events. As a matter of fact, medium- or large-size embedded systems usually require both types of sensors and actuators to be used. On top of that, in complex systems, different control subtasks have to be performed concurrently since the key to achieve successful designs is to apply the "divide and conquer" approach to the global system, in order to break down its functionality into smaller, simpler subsystems.

As one might think, the previous paragraphs may serve as introductory section for a book on any type of embedded systems, these being based on microcontrollers, computers, application-specific integrated circuits (ASICs), or (of course) FPGAs. Therefore, since implementation platforms do not actually modify the earlier definitions and discussion significantly, one of the main objectives of this book is to show when and how FPGAs could (or should) be used for the efficient implementation of embedded control systems targeting industrial applications. Since each technology has its own advantages and limitations, decision criteria must be defined to select

the technology or technologies best suited to solve a given problem. Fairly speaking, the authors do not claim FPGAs to be used for any industrial control system, but their intention is to help designers identify the cases where FPGA technology provides advantages (or is the only possibility) for the implementation of embedded systems in a particular application or application domain.

## 1.3 Implementation Options for Embedded Systems

Selecting the most suitable technique to implement an embedded system that fulfills all the requirements of a given application may not be a trivial issue since designers need to consider many different interrelated factors. Among the most important ones are cost, performance, energy consumption, available resources (i.e., computing resources, sizes of different types of memories, or the number and type of inputs and outputs available), reliability and fault tolerance, or availability. Even if these are most likely the factors with higher impact on design decision, many others may also be significant in certain applications: I/O signal compatibility, noise immunity (which is strongly application dependent), harsh environmental operating conditions (such as extreme temperature or humidity), tolerance to radiations, physical size restrictions, special packaging needs, availability of the main computing device and/or of companion devices (specific power supplies, external crystal oscillators, specific interfaces, etc.), existence of second sources of manufacturing, time to product deprecation, intellectual property (IP) protection, and so forth.

For simple embedded systems, small microcontrollers and small FPGAs are the main market players. As the complexity of the applications to be supported by the embedded system grows, larger FPGAs have different opponents, such as digital signal processing (DSP) processors, multicore processors, general-purpose graphic processing units (GPGPUs), and ASICs. In order to place the benefits and drawbacks of FPGAs within this contest, qualitative and quantitative comparisons between all these technologies are presented in the next sections for readers to have sound decision criteria to help determine what are the most appropriate technologies and solutions for a given application.

### 1.3.1 Technological Improvements and Complexity Growth

The continuous improvements in silicon semiconductor fabrication technologies (mainly resulting in reductions of both transistor size and power supply voltage) implicitly allow lower energy consumption and higher performance to be achieved. Transistor size reduction also opens the door for

higher integration levels, which although undoubtedly being a big advantage for designers, give rise to some serious threats regarding, for instance, circuit reliability, manufacturing yield (i.e., the percentage of fabricated parts that work correctly), or noise immunity.

Power consumption is also becoming one of the main problems faced by designers, not only because of consumption itself but also because of the need for dissipating the resulting heat produced in silicon (especially with modern 3D stacking technologies, where different silicon dies are decked, reducing the dissipation area while increasing the number of transistors—and, therefore, the power consumption—per volume unit). Circuits at the edge of the technology are rapidly approaching the limits in this regard, which are estimated to be around 90 W/cm$^2$, according to the challenges reported for reconfigurable computing by the High Performance and Embedded Architectures and Computation Network of Excellence (www.hipeac.net).

The integration capacity is at the risk of Moore's law starting to suffer from some fatigue. As a consequence, the continuous growth of resource integration over the years is slowing down compared to what has been happening over the last few decades. Transistor sizes are not being reduced at the same pace as higher computing performance is being demanded, so larger circuits are required. Larger circuits negatively affect manufacturing yield and are negatively affected by process variation (e.g., causing more differences to exist between the fastest and slowest circuits coming out from the same manufacturing run, or even having different parts of the same circuit achieving different maximum operating frequencies). This fact, combined with the use of lower power supply voltages, also decreases fault tolerance, which therefore needs to be mitigated by using complex design techniques and contributes to a reduction in system lifetime.

Maximum operating frequency seems to be saturated in practice. A limit of a few GHz for clock frequencies has been reached in regular CMOS technologies with the smallest transistor sizes, no matter the efforts of circuit designers to produce faster circuits, for instance, by heavily pipelining their designs to reduce propagation delay times of logic signals between flip-flops (the design factor that, apart from the technology itself, limits operating frequency).

Is the coming situation that critical? Probably not. These problems have been anticipated by experts in industry and academia, and different approaches are emerging to handle most of the issues of concern. For instance, low-power design techniques are reaching limits that were not imaginable 10–15 years ago, using dynamic voltage scaling or power gating and taking advantage of enhancements in transistor threshold voltages (e.g., thanks to multithreshold transistors). Anyway, the demand for higher computing power with less energy consumption is still there. Mobile devices have a tremendous, ever-increasing penetration in all aspects of our daily lives, and the push of all these systems is much higher than what technology, alone, can handle.

Is there any possibility to face these challenges? Yes, using parallelism. Computer architectures based on single-core processors are no longer providing better performance. Different smart uses of parallel processing are leading the main trends in HPC, as can be seen in the discussion by Kaeli and Akodes (2011). Actually, strictly speaking, taking the most advantage of parallelism does not just mean achieving the highest possible performance using almost unlimited computing resources but also achieving the best possible performance–resources and performance–energy trade-offs. This is the goal in the area of embedded systems, where resources and the energy budget are limited.

In this context, hardware-based systems and, in particular, configurable devices are opening new ways for designing efficient embedded systems. Efficiency may be roughly measured by the ratio of the *number of operations per unit of energy consumed*, for example, in MFlops/mW (millions of floating-point operations per second per milliwatt). Many experiments have shown that improvements of two orders of magnitude may be achieved by replacing single processors with hardware computing.

### 1.3.2 Toward Energy-Efficient Improved Computing Performance

FPGAs have an increasingly significant role when dealing with energy consumption. Sometime ago, the discussion regarding consumption would have been centered on *power* consumption, but the shift toward considering also *energy* as a major key element comes from the fact that the concern on energy availability is becoming a global issue. Initially, it just affected portable devices or, in a more general sense, battery-operated devices with limited usability because of the need of recharging or replacing batteries.

However, the issue of energy usage in any computing system is much more widespread. The most opposite case to restricted-energy, restricted-resource tiny computing devices might be that of huge supercomputing centers (such as for cloud-computing service providers). The concern on the "electricity bill" of these companies is higher than ever. To this respect, although FPGAs cannot be considered the key players in this area, they are presently having a growing penetration. As a proof of that, it can be noticed that there are services (including some any of us could be using daily, such as web searchers or social network applications) currently being served by systems including thousands of FPGAs instead of thousands of microprocessors.

Why is this happening? Things have changed in recent years, as technologies and classical computing architectures are considered to be mature enough. There are some facts (slowly) triggering a shift from software-based to hardware-based computing. As discussed in Section 1.3.1, fabrication technologies are limited in the maximum achievable clock speed, so no more computing performance can be obtained from this side. Also, single-core microprocessor architectures have limited room for enhancement. Even considering complex cache or deep pipelined structures, longer data size

operators, or other advanced features one might think of, there are not much chances for significant improvements.

There is no way of significantly improving performance in a computing system other than achieving higher levels of parallelism. To this respect, the trend in software-based computing is to move from single-core architectures to multi- or many-core approaches. By just taking into account that hardware-based computing structures (and, more specifically, FPGAs) are intrinsically parallel, it means that FPGAs have better chances than software-based computing solutions to provide improved computing performance (Jones et al. 2010).

In addition, if the energy issue is included in the equation, it is clear that reducing the time required to perform a given computation helps in reducing the total energy consumption associated with it (moreover, considering that, in most applications, dynamic power consumption is dominant over static power consumption*). Thus, a certain device with higher power consumption than an opponent device may require less energy to complete a computing task if the acceleration it provides with respect to the opponent is sufficiently high. This is the case with FPGAs, where the computing fabric can be tailored to exploit the best achievable parallelism for a given task (or set of tasks), having in mind both acceleration and energy consumption features.

### 1.3.3 A Battle for the Target Technology?

The graph in Figure 1.2 qualitatively shows the performance and flexibility offered by the different software- and hardware-based architectures suitable for embedded system design. Flexibility is somewhat related to the ease of use of a system and its possibilities to be adapted to changes in specifications.

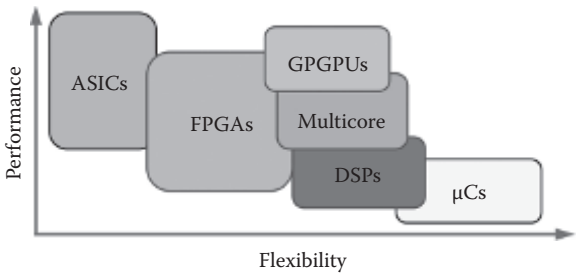


**FIGURE 1.2**
Performance versus flexibility of different technologies for embedded system design.

* Even though, since technologies with higher integration levels and higher resource availability suffer more from static power consumption, this factor needs to be taken into consideration.

As can be seen from Figure 1.2, the highest performance is achieved by ASICs and GPGPUs. The lack of flexibility of ASICs is compensated by their very high energy efficiency. In contrast, although GPGPUs are excellent (if not the best) performant software-based computing devices, they are highly power consuming. Multicore technologies are close to GPGPUs in performance, and, in some cases, their inherent parallelism matches better the one required by the target application. While GPGPUs exploit data parallelism more efficiently, multicore systems are best suited to multitask parallelism. Known drawbacks of GPGPUs and many multicore systems include the need for relying on a host system and limited flexibility with respect to I/O availability. At the high end of flexibility, DSP processors offer better performance than single general-purpose microprocessors or microcontrollers because of their specialization in signal processing. This, of course, is closely related to the amount of signal processing required by the target application.

FPGAs are represented as less flexible than specialized and general-purpose processors. This comes from the fact that software-based solutions are in principle more flexible than hardware-based ones. However, there exist FPGAs that can be reconfigured during operation, which may ideally be regarded to be as flexible as software-based platforms since reconfiguring an FPGA essentially means writing new values in its configuration memory. This is very similar to modifying a program in software approaches, which essentially means writing the new code in program memory. Therefore, FPGAs can be considered in between ASICs and software-based systems, in the sense that they have hardware-equivalent speeds with software-equivalent flexibility. Shuai et al. (2008) provide a good discussion on FPGAs and GPUs.

Figure 1.3 shows a comparative diagram of the aforementioned approaches. The legend shows the axes for flexibility, power efficiency, performance, unit cost, and design cost (complexity).

The cost associated with design complexity is important for embedded devices because the number of systems to be produced may not be too large. Since in the area of embedded systems, significant customization and design effort are needed for every design, additional knowledge is demanded as complexity grows. Hence, complex systems might require a lot of design expertise, which is not always available in small- or medium-sized design offices and labs. Design techniques and tools are therefore very important in embedded system design. They are briefly analyzed in Section 1.3.4. Moreover, tools related to FPGA-based design are analyzed in detail in Chapter 6.

### 1.3.4 Design Techniques and Tools for the Different Technologies

Some design techniques and tools (e.g., those related to PCB design and manufacturing) are of general applicability to all technologies mentioned so far. According to the complexity of the design, these might include techniques and tools for electromagnetic protection and emission mitigation,

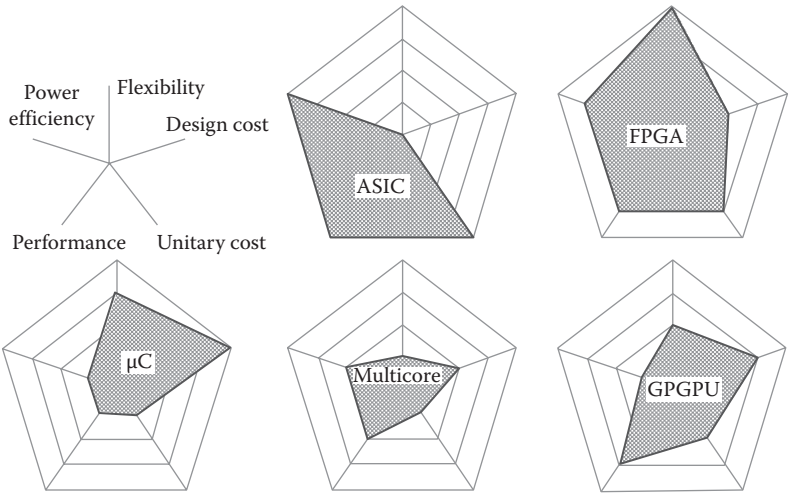


**FIGURE 1.3**
Comparative features of ASICs, FPGAs, general-purpose processors and microcontrollers, multicore processors, and GPGPUs. *Notes:* Outer (further away from the center) is better. ASIC cost applies to large enough production.

thermal analysis, signal integrity tests, and simulation. Some other techniques and tools are specific to the particular technology used (or to some but not all of them), as discussed in the following sections.

#### *1.3.4.1 General-Purpose Processors and Microcontrollers*

General-purpose processors and microcontrollers are the best (and sometimes the only practically viable) solution for simple embedded systems with low computing power requirements. They just require designers' knowledge in programming and debugging simple programs, mostly (if not totally) written in high-level languages. Emulators and debuggers help in validating the developed programs, contributing to a fast design cycle.

In the simplest cases, no operating system (OS) is required; the processor just runs the application program(s), resulting in the so-called bare-metal system. As complexity grows, applications might require complex use of interrupts or other processor features, therefore making it necessary to use an OS as a supporting layer for running multiple tasks concurrently. In this case, dead times of a task can be used to run other tasks, giving some impression of parallelism (although actually it is just concurrency). In very specific cases, system requirements might demand the use of assembly code in order to accelerate some critical functions.

In brief, off-the-shelf simple and cheap processors must be used whenever they are powerful enough to comply with the requirements of simple target applications. Other platforms should be considered only if they provide a

significant added value. For instance, it may be worth using FPGAs to solve a simple application if, in addition, they implement glue logic and avoid the need for populating PCBs with annoying discrete devices required just for interconnection or interfacing purposes.

#### *1.3.4.2 DSP Processors*

Because of their specific architectures, DSP processors are more suitable than general-purpose processors or microcontrollers for many applications above a certain complexity level, where they provide better overall characteristics when jointly considering performance, cost, and power consumption. For instance, DSP-based platforms can solve some problems by working at lower clock frequencies than general-purpose processors would require (therefore reducing energy consumption) or achieving higher throughput (if they work at the same clock frequency).

Specific DSP features are intended to increase program execution efficiency. These include hardware-controlled loops, specialized addressing modes (e.g., bit reverse, which dramatically reduces execution times in fast Fourier transforms), multiply–accumulate (MAC) units (widely used in many signal processing algorithms), multiple memory banks for parallel access to data, direct memory access (DMA) schemes for automated fast I/O, and the ability to execute some instructions in a single clock cycle. There are complex issues (both hardware and software) related to the efficient use of these features. For instance, real parallel operation of functional units can be achieved by using very long instruction word (VLIW) architectures. VLIW systems are capable of determining (at compile time) whether several functional units can be used simultaneously (i.e., if at a given point of program execution there are instructions requiring different functional units to carry out operations with different data that are already available). This implies the ability of the hardware to simultaneously access several memories and fetch several operands or coefficients, which are simultaneously sent to several functional units. In this way, by using multiple MAC units, it may be possible, for instance, to compute several stages of a filter in just one clock cycle.

Advantage can only be taken from DSP processors with VLIW architectures with deep knowledge of the architecture and carefully developing assembly code for critical tasks. Otherwise, they may be underused, and performance could even be worse than when using more standard architectures.

For those embedded systems where the performance of off-the-shelf DSP processors complies with the requirements of the application and that of general-purpose processors or microcontrollers does not, the former are in general the best implementation platform. However, like in the case of general-purpose processors and microcontrollers, except for some devices tailored for specific applications, it is not unusual that external acquisition circuitry is required (e.g., for adaptation to specific I/O needs), which may justify the use of FPGAs instead of DSP processors.

Although DSP processors exploit parallelism at functional module level, it might be the case that the maximum performance they offer is not enough for a given application. In this case, real parallel platforms need to be used. Software-based parallel solutions (multicore processors and GPGPUs) are discussed in Section 1.3.4.3 and hardware-based ones (FPGAs) in Section 1.3.4.4.

#### *1.3.4.3 Multicore Processors and GPGPUs*

The internal architectures of multicore processors and GPGPUs are designed to match task parallelism and data parallelism, respectively. Multicore systems can very efficiently execute multiple, relatively independent tasks, which are distributed among a network of processing cores, each of them solving either a different task or some concurrent tasks. GPGPUs contain a large number of computing elements executing threads that, in a simplistic manner, may be considered as relatively (but not fully) independent executions of the same code over different pieces of data.

Multicore devices can be programmed using conventional high-level languages (such as C or C++), just taking into consideration that different portions of the code (i.e., different tasks) are to be assigned to different processors. The main issues regarding the design with these platforms are related to the need for synchronization and data transfer among tasks, which are usually addressed by using techniques such as semaphores or barriers, when the cores share the same memory, or with message passing through interconnection networks, when each core has its own memory. These techniques are quite complex to implement (in particular for shared memory systems) and also require detailed, complex debugging. For systems with hard real-time constraints, ensuring the execution of multiple tasks within the target deadlines becomes very challenging, although some networking topologies and resource management techniques can help in addressing, to a certain extent, the predictability problem (not easily, though).

GPGPUs operate as accelerators of (multi)processor cores (hosts). The host runs a sequential program that, at some point in the execution process, launches a kernel consisting of multiple threads to be run in the companion GPGPU. These threads are organized in groups, such that all threads within the same group share data among them, whereas groups are considered to be independent from each other. This allows groups to be executed in parallel according to the execution units available within the GPGPU, resulting in the so-called virtual scalability. Thread grouping is not trivial, and the success of a heavily accelerated algorithm depends on groups efficiently performing memory accesses. Otherwise, kernels may execute correctly but with low performance gains (or even performance degradation) because of the time spent in data transfers to/from GPGPUs from/to hosts. Programming kernels requires the use of languages with explicit parallelism, such as CUDA or OpenCL. Debugging is particularly critical (and nontrivial) since a careful and detailed analysis is required to prevent malfunctions caused

by desynchronization of threads, wrong memory coalescence policies, or inefficient kernel mapping. Therefore, in order for GPGPUs to provide better performance than the previously discussed implementation platforms, designers must have deep expertise in kernel technology and its mapping in GPGPU architectures.

#### *1.3.4.4 FPGAs*

FPGAs offer the possibility of developing configurable, custom hardware that might accelerate execution while providing energy savings. In addition, thanks to the increasing scale of integration provided by fabrication technologies, they can include, as discussed in detail in Chapter 3, one or more processing cores, resulting in the so-called field-programmable systems-on-chip (FPSoCs) or systems-on-programmable chip (SoPCs).* These systems may advantageously complement and extend the characteristics of the aforementioned single- or multicore platforms with custom hardware accelerators, which allow the execution of all or some critical tasks to be optimized, both in terms of performance and energy consumption.

Powerful design tools are required to deal with the efficient integration of these custom hardware peripherals and others from off-the-shelf libraries, as well as other user-defined custom logic, with (multiple) processor cores in an FPSoC architecture. These SoPC design tools, described in Chapter 6, require designers to have good knowledge of the underlying technologies and the relationship among the different functionalities needed for the design of FPSoCs, in spite of the fact that vendors are making significant efforts for the automation and integration of all their tools in single (but very complex) environments. In this sense, in the last few years, FPGA vendors are offering solutions for multithreaded acceleration that compete with GPGPUs, thus providing tools to specify OpenCL kernels that can be mapped into FPGAs. Also, long-awaited high-level synthesis (HLS) tools now provide a method to migrate from high-level languages such as C, C++, or SystemC into hardware description languages (HDLs), such as VHDL or Verilog, which are the most widely used today by FPGA design tools.

#### *1.3.4.5 ASICs*

ASICs are custom integrated circuits (mainly nonconfigurable, in the sense explained in Section 1.4) fully tailored for a given application. A careful design using the appropriate manufacturing technology may yield excellent performance and energy efficiency, but the extremely high nonrecurrent engineering costs and the very specific and complex skills required to design them make this target technology unaffordable for low- and medium-sized productions.

* Since the two acronyms may be extensively found in the literature as well as in vendor-provided information, both of them are interchangeably used throughout this book.

The lack of flexibility is also a problem, since, nowadays, many embedded systems need to be capable of being adapted to very diverse applications and working environments. For instance, the ability to adapt to changing communication protocols is an important requirement in many industrial applications.

## 1.4 How Does Configurable Logic Work?

First of all, it is important to highlight the intrinsic difference between *programmable* and (*re*)*configurable* systems. The "P" in FPGA can be misleading since, although FPGAs are the most popular and widely used *configurable* circuits, it stands for *programmable*. Both kinds of systems are intended to allow users to change their functionality. However, not only in the context of this book but also according to most of the literature and the specialized jargon, *programmable* systems (processors) are those based on the execution of software, whereas (*re*)*configurable* systems are those whose internal hardware computing resources and interconnects are not totally configured by default. Configuration consists in choosing, configuring, and interconnecting the resources to be used. Software-based solutions typically rely on devices whose hardware processing structure is fixed, although, as discussed in Chapter 3, the configurable hardware resources of an FPGA can be used to implement a processor, which can then obviously be programmed.

The fixed structure of programmable systems is built so as to allow them to execute different sequences (software programs) of basic operations (instructions). The programming process mainly consists in choosing the right instructions and sequences for the target application. During execution, instructions are sequentially fetched from memory, then (if required) data are fetched from memory or from registers, the processing operation implied by the current instruction is computed, and the resulting data (if any) are written back to memory or registers. As can be inferred, the hardware of these systems does not provide functionality by itself, but through the instructions that build up the program being executed.

On the other hand, in configurable circuits, the structure of the hardware resources resulting from the configuration of the device determines the functionality of the system. Using different configurations, the same device may exhibit different internal functional structures and, therefore, different user-defined functionalities. The main advantage of configurable systems with regard to pure software-based solutions is that, instead of sequentially executing instructions, hardware blocks can work in a collaborative concurrent way; that is, their execution of tasks is inherently parallel.

Arranging the internal hardware resources to implement a variety of digital functions is equivalent, from a functional point of view, to manufacturing different devices for different functions, but with configurable circuits,

no further fabrication steps are required to be applied to the premanufactured off-the-shelf devices. In addition, configuration can be done at the user premises, or even infield at the operating place of the system.

The beginning of reconfigurable devices started with programmable* logic matrices (programmable logic array [PLA] and programmable array logic [PAL]—whose basic structures are shown in Figure 1.4), where the connectivity of signals was decided using arrays of programmable connections. These were originally fuses (or antifuses†), which were selectively either burnt or left intact during configuration.

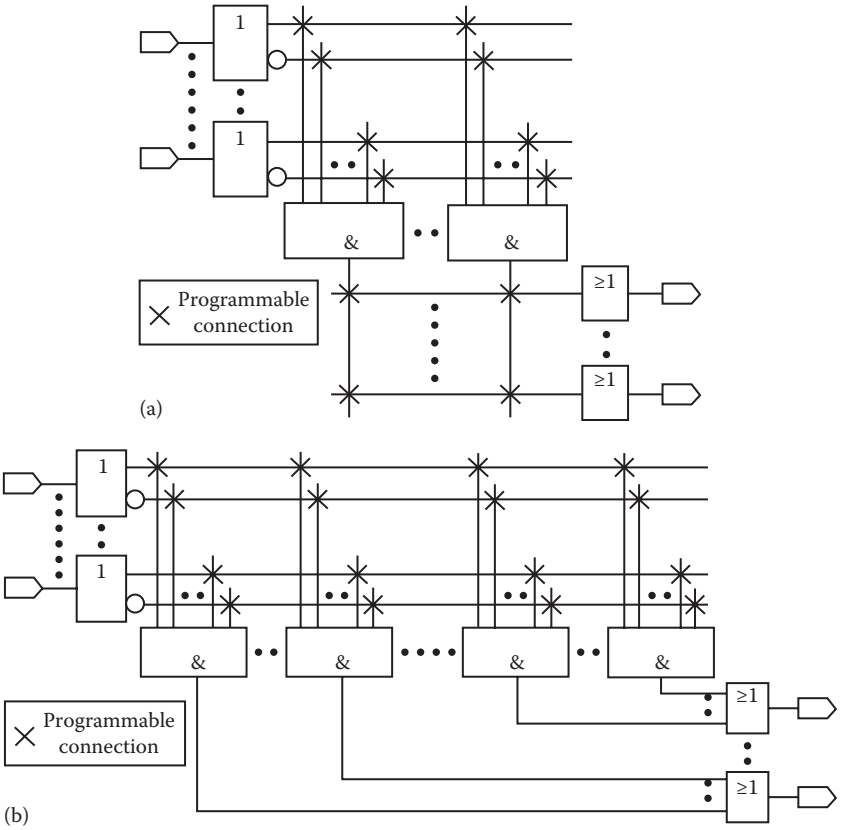


**FIGURE 1.4**
Programmable matrices: (a) PLA; (b) PAL.

* At that time, the need for differentiating programmability and configurability had not yet been identified.

† The difference between fuses and antifuses resides in their state after being burnt, open or short circuit, respectively.

In programmable matrices, configuration makes the appropriate input signals participate in the sums of products required to implement different logic functions. When using fuses, this was accomplished by selectively overheating those to be burnt, driving a high current through them. In this case, the structural internal modifications are literally real and final, since burnt fuses cannot be configured back to their initial state.

Although the scale of integration of fuses was in the range of several micrometers (great for those old days), CMOS integration was advancing at a much faster pace, and quite soon, new configuration infrastructures were developed in the race for larger, faster, and more flexible reconfigurable devices. Configuration is no longer based on changes in the physical structure of the devices, but on the behavior regarding connectivity and functionality, specified by the information stored in dedicated memory elements (the so-called *configuration* memory). This not only resulted in higher integration levels but also increased flexibility in the design process, since configurable devices evolved from being one-time programmable to being reconfigurable, which can be configured several times by using erasable and reprogrammable memories for configuration. Nowadays, a clear technological division can be made between devices using nonvolatile configuration memories (EEPROM and, more recently, flash) and those using volatile configuration memories (SRAM, which is the most widely used technology for FPGA configuration).

Currently, programmable matrices can be found in programmable logic devices (PLDs), which found their application niche in glue logic and finite-state machines. The basic structure of PLDs is shown in Figure 1.5. In addition to configuring the connections between rows and columns of the programmable matrices, in PLDs, it is also possible to configure the behavior of the macrocells.

The main drawback of PLDs comes from the scalability problems related to the use of programmable matrices. This issue was first addressed by including several PLDs in the same chip, giving rise to the complex PLD concept. However, it soon became apparent that this approach does not solve the scalability problem to the extent required by the ever-increasing complexity of digital systems, driven by the evolution of fabrication technologies. A change in the way configurable devices were conceived was needed. The response to that need were FPGAs, whose basic structure is briefly described in the following.*

Like all configurable devices, FPGAs are premanufactured, fixed pieces of silicon. In addition to configuration memory, they contain a large number of basic configurable elements, ideally allowing them to implement any digital system (within the limits of the available chip resources). There are two main types of building blocks in FPGAs: (relatively small) configurable logic circuits spread around the whole chip area (logic blocks [LBs]) and, between them, configurable interconnection resources (interconnect logic [IL]).

* FPGA architectures are analyzed in detail in Chapter 2.

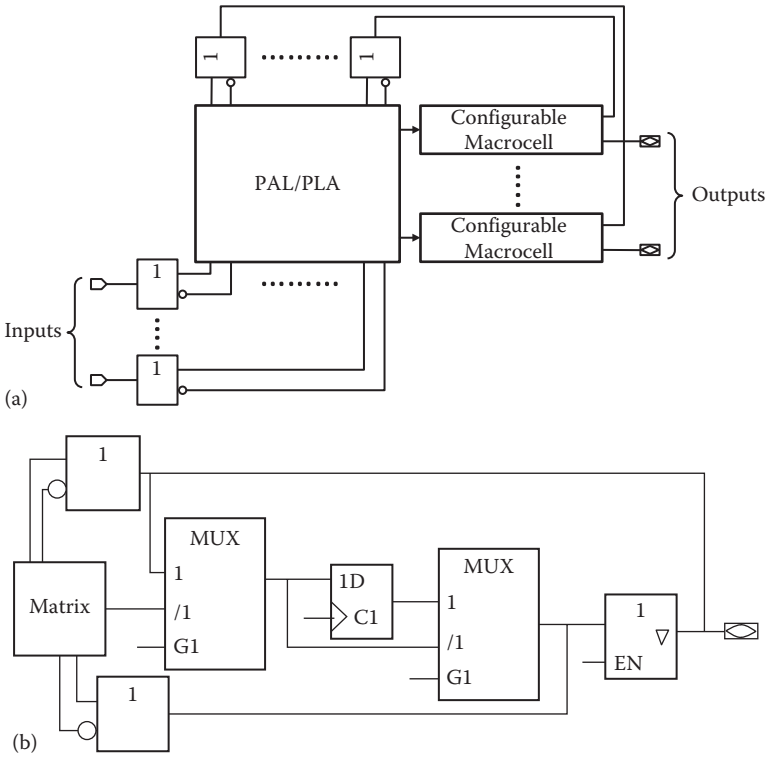


**FIGURE 1.5**
(a) Basic PLD structure; (b) sample basic macrocell.

The functionality of the target system is obtained by adequately configuring the behavior of the required LBs and the IL that interconnects them, by writing the corresponding information in the FPGA's internal configuration memory. The information is organized in the form of a stream of binary data (called bitstream) coming out from the design process, which determines the behavior of every LB and every interconnection inside the device. FPGA configuration issues are analyzed in Chapter 6.

A most basic LB would consist of the following:

- A small SRAM memory ($2^n \times 1$, with a value of $n$ typically from 4 to 6) working as a lookup table (LUT), which allows any combinational function of its $n$ inputs to be implemented. A LUT can be thought of as a way of storing the truth table of the combinational function in such a way that, when using the inputs of that function as address bits of the LUT, the memory bit storing the value of the function for each particular input combination can be read at the output of the LUT.

- A flip-flop whose data input is connected to the output of the LUT.
- A multiplexer (MUX) that selects as output of the LB either the flip-flop output or the LUT output (i.e., the flip-flop input). In this way, depending on the configuration of the MUX, the LB can implement either combinational or sequential functions.
- The inputs of the LB (i.e., of the LUT) and its output (i.e., of the MUX), connected to nearby IL.

In practice, actual LBs consist of a combination of several (usually two) of these basic LUT/flip-flop/MUX blocks (which are sometimes referred to as slices). They also often include specific logic to generate and propagate carry signals (both inside the LB itself and between neighbor LBs, using local carry-in and carry-out connections), resulting in the structure shown in Figure 1.6. Typically, in addition, the LUTs inside an LB can be combined to form a larger one, allowing combinational functions with a higher number of inputs to be implemented, thus providing designers with extra flexibility.

In addition, current FPGAs also include different kinds of specialized resources (described in detail in Chapters 2 through 5), such as memories and memory controllers, DSP blocks (e.g., MAC units), and embedded processors and commonly used peripherals (e.g., serial communication interfaces), among others. They are just mentioned here in order for readers to understand the ever-increasing application scope of FPGAs in a large variety of industrial control systems, some of which are highlighted in Section 1.5 to conclude this chapter.

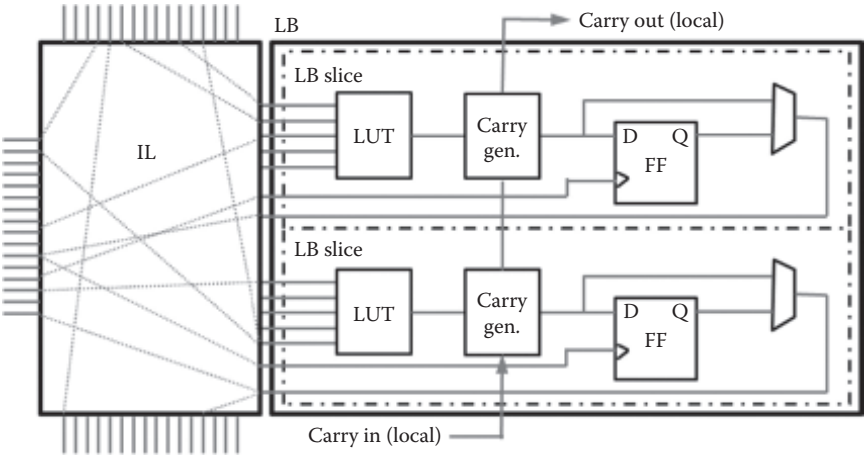


**FIGURE 1.6**
Example of two-slice LB and its connection to IL.

## 1.5 Applications and Uses of FPGAs

The evolution from "traditional" FPGA architectures, mainly consisting of basic standard reconfigurable building blocks (LBs and IL), to more feature-rich, heterogeneous devices is widening the fields of applicability of FPGAs, taking advantage of their current ability to implement entire complex systems in a single chip. FPGAs are not used anymore just for glue logic or emulation purposes, but have also fairly gained their own position as suitable platforms to deal with increasingly complex control tasks and are also getting, at a very fast pace, into the world of HPC.

This technological trend has also extended the applicability of FPGAs in their original application domains. For instance, emulation techniques are evolving into mixed solutions, where the behavior of (parts of) a system can be evaluated by combining simulation models with hardware emulation, in what is nowadays referred to as hardware-in-the-loop (HIL). Tools exist, including some of general use in engineering, such as MATLAB®, which allow this combined simulation/emulation approach to be used to accelerate system validation.

FPGAs are also increasingly penetrating the area of embedded control systems, because in many cases, they are the most suitable solution to deal with the growing complexity problems to be addressed in that area. Some important fields of application (not only in terms of technological challenges but also in terms of digital systems' market share) are in automated manufacturing, robotics, control of power converters, motion and machinery control, and embedded units in automotive (and all transportation areas in general)—it is worth noting that a modern car has some 70–100 embedded control units onboard. As the complexity of the systems to be controlled grows, microcontroller and DSPs are becoming less and less suitable, and FPGAs are taking the floor.

A clear proof of the excellent capabilities of current FPGAs is their recent penetration in the area of HPC, where a few years ago, no one would have thought they could compete with software approaches implemented in large processor clusters. However, computing-intensive areas such as big data applications, astronomical computations, weather forecast, financial risk management, complex 3D imaging (e.g., in architecture, movies, virtual reality, or video games), traffic prediction, earthquake detection, and automated manufacturing may currently benefit from the acceleration and energy-efficient characteristics of FPGAs.

One may argue these are not typical applications of industrial embedded systems. There is, however, an increasing need for embedded high-performance systems, for example, systems that must combine intensive computation capabilities with the requirements of embedded devices, such as portability, small size, and low-energy consumption. Examples of such applications are complex wearable systems in the range of augmented or

virtual reality, automated driving vehicles, and complex vision systems for robots or in industrial plants. The Internet of Things is one of the main forces behind the trend to integrate increasing computing power into smaller and energy-efficient devices, and FPGAs can play an important role in this scenario.

Given the complexity of current devices, FPGA designers have to deal with many different issues related to hardware (digital and analog circuits), software (OSs and programming for single- and multicore platforms), tools and languages (such as HDLs, C, C++, SystemC, as well as some with explicit parallelism, such as CUDA or OpenCL), specific design techniques, and knowledge in very diverse areas such as control theory, communications, and signal processing. All these together seem to point to the need for super-engineers (or even super-engineering teams), but do not panic. While it is not possible to address all these issues in detail in a single book, this one intends at least to point industrial electronics professionals who are not specialists in FPGAs to the specific issues related to their working area so that they can first identify them and then tailor and optimize the learning effort to fulfill their actual needs.

## References

Jones, D.H., Powell, A., Bouganis, Ch.-S., and Cheung, P.Y.K. 2010. GPU versus FPGA for high productivity computing. In *Proceedings of the 20th International Conference on Field Programmable Logic and Applications*, August 31 to September 2, Milano, Italy.

Kaeli, D. and Akodes, D. 2011. The convergence of HPC and embedded systems in our heterogeneous computing future. In *Proceedings of the IEEE 29th International Conference on Computer Design (ICCD)*, October 9–12, Amherst, MA.

Laprie, J.C. 1985. Dependable computing and fault tolerance: Concepts and terminology. In *Proceedings of the 15th Annual International Symposium on Fault-Tolerant Computing (FTCS-15)*, June 19–21, Ann Arbor, MI.

Shuai, C., Jie, L., Sheaffer, J.W., Skadron, K., and Lach, J. 2008. Accelerating compute-intensive applications with GPUs and FPGAs. In *Proceedings of the Symposium on Application Specific Processors (SASP 2008)*, June 8–9, Anaheim, CA.

## FPGAs and Their Role in the Design of Electronic Systems

Jones, D.H. , Powell, A. , Bouganis, Ch.-S. , and Cheung, P.Y.K. 2010. GPU versus FPGA for high productivity computing. In Proceedings of the 20th International Conference on Field Programmable Logic and Applications, August 31 to September 2, Milano, Italy.

Kaeli, D. and Akodes, D. 2011. The convergence of HPC and embedded systems in our heterogeneous computing future. In Proceedings of the IEEE 29th International Conference on Computer Design (ICCD), October 9–12, Amherst, MA.

Laprie, J.C. 1985. Dependable computing and fault tolerance: Concepts and terminology. In Proceedings of the 15th Annual International Symposium on Fault-Tolerant Computing (FTCS-15), June 19–21, Ann Arbor, MI.

Shuai, C. , Jie, L. , Sheaffer, J.W. , Skadron, K. , and Lach, J. 2008. Accelerating compute-intensive applications with GPUs and FPGAs. In Proceedings of the Symposium on Application Specific Processors (SASP 2008), June 8–9, Anaheim, CA.

## Main Architectures and Hardware Resources of FPGAs

Achronix . 2008. Introduction to Achronix FPGAs. White paper WP001-1.6.

Achronix . 2015. Speedster22i HD1000 FPGA data sheet DS005-1.0.

Actel (currently Microsemi) . 2010. *ProASIC3 FPGA Fabric User's Guide*.

Altera . 2012. *Cyclone III Device Handbook*.

Altera . 2014. *Stratix V Device Handbook. Vol. 2: Transceivers*.

Altera . 2015a. MAX 10 FPGA device architecture.

Altera . 2015b. *Arria 10 Core Fabric and General Purpose I/Os Handbook*.

Altera . 2015c. *Stratix V Device Handbook. Vol. 1: Device Interfaces and Integration*.

Altera . 2015d. *Arria 10 Transceiver PHY User Guide UG-01143*.

Altera . 2016. *External Memory Interface Handbook Volume 1: Altera Memory Solution Overview, Design Flow, and General Information*.

Barrett, C. 1999. Fractional/integer-N PLL basics. Texas Instruments technical brief SWRA029. Texas Instruments, Dallas, TX.

Cortina Systems and Cisco Systems . 2008. Interlaken protocol definition. Revision 1.2.

Curd, D. 2012. PCI express for the 7 series FPGAs. Xilinx white paper WP384 (v1.1).

Gentile, K. 2008. Introduction to zero-delay clock timing techniques. Analog Devices application note AN-0983. Analog Devices, Norwood, MA.

Hutton, M. 2015. Understanding how the new HyperFlex architecture enables next-generation high-performance systems. Altera white paper WP-01231-1.0.

Jiao, B. 2015. Leveraging UltraScale FPGA transceivers for high-speed serial I/O connectivity. Xilinx white paper WP458 (v1.1).

Kuon, I. , Tessier, R. , and Rose, J. 2007. FPGA architecture: Survey and challenges. Foundations and Trends in Electronic Design Automation, 2:135–253.

Lattice . 2015. iCE40 Ultra family datasheet DS1048 (v1.8).

Lattice . 2016. MachXO3 family datasheet DS1047 (v1.6).

Microsemi . 2014. Fusion family of mixed signal FPGAs datasheet. Revision 6.

Microsemi . 2015a. IGLOO2 FPGA and SmartFusion2 SoC FPGA: Datasheet DS0451.

Microsemi . 2015b. SmartFusion2 SoC and IGLOO2 FPGA fabric: User guide UG0445.

Microsemi . 2015c. ProASIC3E flash family FPGAs: Datasheet DS0098.

Microsemi . 2015d. SmartFusion2 and IGLOO2 clocking resources: User guide UG0449.

PCI-SIG . 2015. PCI Express® base specification revision 3.1a. Available at: https://pcisig.com/specifications/pciexpress. Accessed November 20, 2016 .

QuickLogic . 2010. ArcticLink solution platform datasheet (rev. M).

QuickLogic . 2013. ArcticLink II VX2 solution platform datasheet (rev. 1.0).

Rodriguez-Andina, J.J. , Moure, M.J. , and Valdes, M.D. 2007. Features, design tools, and application domains of FPGAs. IEEE Transactions on Industrial Electronics, 54:1810–1823.

Rodriguez-Andina, J.J. , Valdes, M.D. , and Moure, M.J. 2015. Advanced features and industrial applications of FPGAs—A review. IEEE Transactions on Industrial Informatics, 11:853–864.

Saban, K. 2012. Xilinx Stacked Silicon Interconnect Technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency. Xilinx white paper WP380 (v1.2).
Texas Instruments . 2008. Fractional N frequency synthesis. Application note AN-1879.
Xilinx . 2004. Celebrating 20 years of innovation. Xcell Journal, 48:14–16.
Xilinx . 2006. Virtex-5 platform FPGA family technical backgrounder.
Xilinx . 2010. *Spartan-6 FPGA Configurable Logic Block: User Guide UG384* (*v1.1*).
Xilinx . 2014a. *Spartan-6 FPGA SelectIO Resources: User Guide UG381* (*v1.6*).
Xilinx . 2014b. *7 Series FPGAs Configurable Logic Block: User Guide UG474* (*v1.7*).
Xilinx . 2014c. *7 Series FPGAs Memory Resources: User Guide UG473* (*v1.11*).
Xilinx . 2014d. *7 Series FPGAs GTP Transceivers: User Guide UG482* (*v1.8*).
Xilinx . 2015a. *7 Series FPGAs SelectIO Resources: User Guide UG471* (*v1.5*).
Xilinx . 2015b. *7 Series FPGAs Clocking Resources: User Guide UG472* (*v1.11.2*).
Xilinx . 2015c. *7 Series FPGAs GTX/GTH Transceivers: User Guide UG476* (*v1.11*).

## Embedded Processors in FPGA Architectures

Altera . 2002. Excalibur device overview data sheet. DS-EXCARM-2.0.
Altera . 2003. Avalon bus specification reference manual. MNL-AVABUSREF-1.2.
Altera . 2013. AMBA AXI and Altera Avalon Interoperation using Qsys. Available at: https://www.youtube.com/watch?v=LdD2B1x-5vo. Accessed November 20, 2016 .
Altera . 2015a. Avalon interface specifications. MNLAVABUSREF 2015.03.04.
Altera . 2015b. Quartus prime standard edition handbook. QPS5V1 2015.05.04.
Altera . 2015c. Nios II classic processor reference guide. NII5V1 2015.04.02.
Altera . 2015d. Stratix 10 device overview data sheet. S10-OVERVIEW.
Altera . 2016a. Arria 10 hard processor system technical reference manual. Available at: https://www.altera.com/en_US/pdfs/literature/hb/arria-10/a10_5v4.pdf. Accessed November 20, 2016 .
Altera . 2016b. Arria 10 device data sheet. A10-DATASHEET.
ARM . 1999. AMBA specification (rev 2.0) datasheet. IHI 0011A.
ARM . 2004a. Multilayer AHB overview datasheet. DVI 0045B.
ARM . 2004b. AMBA AXI protocol specification (v1.0) datasheet. IHI 0022B.
ARM . 2008. Cortex-M1 technical reference manual. DDI 0413D.
ARM . 2011. AMBA AXI and ACE protocol specification datasheet. IHI 0022D.
ARM . 2012. Cortex-A9 MPCore technical reference manual (rev. r4p1). ID091612.
Atmel . 2002. AT94K series field programmable system level integrated circuit data sheet. 1138F-FPSLI-06/02.
Bergamaschi, R.A. and Lee, W.R. 2000. Designing systems-on-chip using cores. In Proceedings of the 37th Design Automation Conference (DAC 2000). June 5–9, Los Angeles, CA.
Cadence . 2014. Tensilica Xtensa 11 customizable processor datasheet.
IBM . 1999. The CoreConnect™ bus architecture.
Jeffers, J. and Reinders, J. 2015. High Performance Parallelism Pearls. Multicore and Many-Core Programming Approaches. Elsevier.
Kalray . 2014. MPPA ManyCore. Available at: http://www.kalrayinc.com/IMG/pdf/FLYER_MPPA_MANYCORE.pdf. Accessed November 20, 2016 .
Kenny, R. and Watt, J. 2016. The breakthrough advantage for FPGAs with tri-gate technology. White Paper WP-01201-1.4. Available at:https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01201-fpga-tri-gate-technology.pdf. Accessed November 23, 2016 .
Kurisu, W. 2015. Addressing design challenges in heterogeneous multicore embedded systems. Mentor Graphics white paper TECH12350-w.
Lattice . 2008. Linux port to LatticeMico32 system reference guide.

Lattice . 2012. LatticeMico32 processor reference manual.
Lattice . 2014. LatticeMico8 processor reference manual.
Microsemi . 2013. SmartFusion2 microcontroller subsystem user guide.
Microsemi . 2016. SmartFusion2 system-on-chip FPGAs product brief. Available at: http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2#documentation. Accessed November 20, 2016 .
Moyer, B. 2013. Real World Multicore Embedded Systems: A Practical Approach. Elsevier–Newnes.
Nickolls, J. and Dally, W.J. 2010. The GPU computing era. IEEE Micro, 30:56–69.
NVIDIA . 2010. NVIDIA Tegra multi-processor architecture. Available at: http://www.nvidia.com/docs/io/90715/tegra_multiprocessor_architecture_white_paper_final_v1.1.pdf. Accessed November 20, 2016 .
OpenCores . 2011. OpenRISC 1200 IP core specification (v0.11).
Pavlo, A. 2015. Emerging hardware trends in large-scale transaction processing. IEEE Internet Computing, 19:68–71.
QuickLogic . 2001. QL901M QuickMIPS data sheet.
QuickLogic . 2010. Customer specific standard product approach enables platform-based design. White paper (rev. F).
QuickLogic . 2015. QuickLogic EOS S3 sensor processing SoC platform brief. Datasheet.
Shalf, J. , Bashor, J. , Patterson, D. , Asanovic, K. , Yelick, K. , Keutzer, K. , and Mattson, T. 2009. The MANYCORE revolution: Will HPC LEAD or FOLLOW? Available at: http://cs.lbl.gov/news-media/news/2009/the-manycore-revolution-will-hpc-lead-or-follow/.
Sharma M. 2014. CoreSight SoC enabling efficient design of custom debug and trace subsystems for complex SoCs. Key steps to create a debug and trace solution for an ARM SoC. ARM White Paper. Available at: https://www.arm.com/files/pdf/building_debug_and_trace_multicore_soc.pdf. Accessed November 20, 2016 .
Singh, V. and Dao, K. 2013. Maximize system performance using Xilinx based AXI4 interconnects. Xilinx white paper WP417.
Stallings, W. 2016. Computer Organization and Architecture. *Designing for Performance*, 10th edn. Pearson Education, UK.
Sundaramoorthy, N. , Rao, N. , and Hill, T. 2010. AXI4 interconnect paves the way to plug-and-play IP. Xilinx white paper WP379.
Synopsys . 2015. DesignWare ARC HS34 processor datasheet.
Tendler, J.M. , Dodson, J.S. , Fields Jr., J.S. , Le, H. , and Sinharoy, B. 2002. POWER4 system microarchitecture. IBM Journal of Research and Development, 46(1):5–25.
Triscend . 2000. Triscend E5 configurable system-on-chip family data sheet.
Triscend . 2001. Triscend A7 configurable system-on-chip platform data sheet.
Vadja, A. 2011. Programming Many-Core Chips. Springer Science + Business Media.
Walls, C. 2014. Selecting an operating system for embedded applications. Mentor Graphics white paper TECH112110-w.
Xilinx . 2008. Virtex-4 FPGA user guide UG070 (v2.6).
Xilinx . 2010a. PowerPC 405 processor block reference guide UG018 (v2.4).
Xilinx . 2010b. Embedded processor block in Virtex-5 FPGAs reference guide UG200 (v1.8).
Xilinx . 2011a. PicoBlaze 8-bit embedded microcontroller user guide UG129.
Xilinx . 2011b. Virtex-II Pro and Virtex-II Pro X platform FPGAs: Complete data sheet DS083 (v5.0).
Xilinx . 2014. Zynq-7000 all programmable SoC technical reference manual UG585 (v1.7).
Xilinx . 2015a. Vivado design suite—AXI reference guide UG1037.
Xilinx . 2015b. Xilinx collaborates with TSMC on 7nm for fourth consecutive generation of all programmable technology leadership and multi-node scaling advantage. Available at http://press.xilinx.com/2015-05-28-Xilinx-Collaborates-with-TSMC-on-7nm-for-Fourth-Consecutive-Generation-of-All-Programmable-Technology-Leadership-and-Multi-node-Scaling-Advantage. Accessed November 23, 2016 .
Xilinx . 2016a. MicroBlaze processor reference guide UG984.
Xilinx . 2016b. Zynq UltraScale+ MPSoC overview data sheet DS891 (v1.1).

## Advanced Signal Processing Resources in FPGAs

Altera .2011. Stratix IV Device Handbook (Vol. 1). DSP Blocks in Stratix IV Devices. San Jose, CA.

Altera .2016. Arria 10 Core Fabric and General Purpose I/Os Handbook A10. San Jose, CA.

IEEE . 2008. 754-2008—IEEE standard for floating-point arithmetic. Revision of ANSI/IEEE Std 754-1985.

Lattice .2016. ECP5 and ECP5-5G family. Data Sheet DS1044 Version 1.6. Portland, OR.

Microsemi .2016. SmartFusion2 SoC and IGLOO2 FPGA Fabric: UG0445 User Guide. Aliso Viejo, CA.

Parker, M. 2014. Understanding peak floating-point performance claims, Altera white paper WP-01222-1.0.

Sinha, U. 2014. Enabling impactful DSP designs on FPGAs with hardened floating-point implementation, Altera white paper WP-01227-1.0.

Xilinx .2011. Spartan-3 Generation FPGA User Guide: Extended Spartan-3A, Spartan-3E, and Spartan-3 FPGA Families: UG331 (v1.8). San Jose, CA.

Xilinx . 2014. 7 Series DSP48E1 Slice. User Guide: UG479 (v1.8). San Jose, CA.

## Mixed-Signal FPGAs

Altera . 2015. *MAX 10 Analog to Digital Converter User Guide: UG-M10ADC*.

Anadigm . 2006. AN13x series. AN23x series. *AnadigmApex dpASP Family User Manual*.

Cypress . 2015. PSoC 5LP: CY8C58LP family datasheet.

Microsemi . 2014a. *SmartFusion Programmable Analog User Guide*.

Microsemi . 2014b. *Fusion Family of Mixed Signal FPGAs* (*revision 6*).

Xilinx . 2011. *Virtex-5 FPGA System Monitor User Guide: UG192* (*v1.7.1*).

Xilinx . 2014. *Virtex-6 FPGA System Monitor User Guide: UG370* (*v1.2*).

Xilinx . 2015. *7 Series FPGAs and Zynq-7000 All Programmable SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide: UG480* (*v1.7*).

## Tools and Methodologies for FPGA-Based Design

Altera . 2013. Implementing FPGA design with the OpenCL standard. White paper WP-01173-3.0.

Cong, J. , Liu, B. , Neuendorffer, S. , Noguera, J. , Vissers, K. , and Zhang, Z. 2011. High-level synthesis for FPGAs: From prototyping to deployment. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 30:473–491.

Riesgo, T. , Torroja, Y. , and de la Torre, E. 1999. Design methodologies based on hardware description languages. IEEE Transactions on Industrial Electronics, 46:3–12.

Sangiovanni-Vincentelli, A. and Martin, G. 2001. Platform-based design and software design methodology for embedded systems. IEEE Design & Test of Computers, 18:23–33.

Xilinx . 2014. The Xilinx SDAccel development environment.

## Off-Chip and In-Chip Communications for FPGA Systems

Athavale, A. and Christensen, C. 2015. High-speed serial I/O made simple. A designer’s guide with FPGA applications. In Xilinx Connectivity Solutions, Product Solutions Marketing/Xilinx Worldwide Marketing Department, San Jose, CA.

Benini, L. and De Micheli, G. 2006. Networks on chip. In The Morgan Kauffman Series in Systems on Silicon, Elsevier.

Lee, H.G. , Chang, N.C.K. , Ogras, U.Y. , and Marculescu, R. 2007. On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. ACM Transactions on Design Automation of Electronic Systems, 12:3, article 23.

## Building Reconfigurable Systems Using Commercial FPGAs

Al-Hashimi, B.M. , ed. 2006. System-on-Chip: Next Generation Electronics. IET Press, London, U.K.

Athanas, P. , Bowen, J. , Dunham, T. , Patterson, C. , Rice, J. , Shelburne, M. , Suris, J. , Bucciero, M. , and Graf, J. 2007. Wires on demand: Run-time communication synthesis for reconfigurable computing. In Proceedings of the International Conference on Field Programmable Logic and Applications (FPL 2007), August 27–29, Amsterdam, the Netherlands.

Becker, J. , Donlin, A. , and Hubner, M. 2007. New tool support and architectures in adaptive reconfigurable computing. In Proceedings of the IFIP International Conference on Very Large Scale Integration 2007, October 15–17, Atlanta, GA.

Beckhoff, C. , Koch, D. , and Torresen, J. 2012. Go ahead: A partial reconfiguration framework. In Proceedings of the 2012 IEEE 20th Annual International Symposium on Field-Programmable Custom Computing Machines, (FCCM 2012), April 29–May 1, 2012, Toronto, Ontario, Canada, pp. 37–44.

Bobda, C. , Majer, M. , Ahmadinia, A. , Haller, T. , Linarth, A. , and Teich, J. 2005. The Erlangen slot machine: Increasing flexibility, in FPGA-based reconfigurable platforms. In Proceedings of the 2005 IEEE International Conference on Field-Programmable Technology, December 11–14, Singapore.

Cervero, T. , Otero, A. , López, S. , de la Torre, E. , Callicó, G.M. , and Riesgo, T. 2016. A scalable H.264/AVC deblocking filter architecture. Journal of Real-Time Image Processing, 12(1):81–105.

Compton, K. and Hauck, S. 2002. Reconfigurable computing: A survey of systems and software. ACM Computing Surveys, 34:171–210.

Gallego, A. , Mora, J. , Otero, J. , de la Torre, E. , and Riesgo, T. 2013. A scalable evolvable hardware processing array. In Proceedings of the 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2013), December 9–11, Cancún, Mexico.

Guccione, S.A. and Levi, D. 1998. XBI: A Java-based interface to FPGA hardware. In Proceedings of SPIE, Configurable Computing: Technology and Applications, Vol. 3526, Boston, MA, pp. 97–102.

Horta, E.L. , Lockwood, J.W. , Taylor, D.E. , and Parlour, D. 2002. Dynamic hardware plugins in an FPGA with partial run-time reconfiguration. In Proceedings of the 39th Design Automation Conference, June 10–14, New Orleans, LA.

Kephart, J.O. and Chess, D.M. 2003. The vision of autonomic computing. Computer, 36:41–50.

Koch, D. , Beckhoff, C. , and Teich, J. 2008. ReCoBus-Builder—A novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs. In Proceedings of the International Conference on Field Programmable Logic and Applications (FPL 2008), September 8–10, Heidelberg, Germany.

Krasteva, Y.E. , de la Torre, E. , and Riesgo, T. 2008. Virtual architectures for partial runtime reconfigurable systems. Application to Network on Chip based SoC emulation. In Proceedings of the 34th Annual Conference of the IEEE Industrial Electronics Society (IECON 2008), November 10–13, Orlando, FL.

Miller, J.F. 2011. Cartesian Genetic Programming, Natural Computing Series 2011. Springer.

Moller, L. , Soares, R. , Carvalho, E. , Grehs, I. , Calazans, N. , and Moraes, F. 2006. Infrastructure for dynamic reconfigurable systems: Choices and trade-offs. In Proceedings of the 19th Annual Symposium on Integrated Circuits and Systems Design, August 28 to September 1, Ouro Preto, Brazil.

Mora, J. , Otero, A. , de la Torre, E. , and Riesgo, T. 2015. Fast and compact evolvable systolic arrays on dynamically reconfigurable FPGAs. In Proceedings of the 10th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC), June 29 to July 1, Bremen, Germany.

Palma, J.C. , de Mello, A.V. , Moller, L. , Moraes, F. , and Calazans, N. 2002. Core communication interface for FPGAs. In Proceedings of the 15th Annual Symposium on Integrated Circuits and Systems Design, September 9–14, Porto Alegre, Brazil.

Pionteck, T. , Koch, R. , and Albrecht, C. 2006. Applying partial reconfiguration to networks-on-chips. In Proceedings of the 16th International Conference on Field-Programmable Logic and Applications, August 28–30, Madrid, Spain.

Rodriguez, A. , Valverde, J. , Castanares, C. , Portilla, J. , de la Torre, E. , and Riesgo, T. 2015. Execution modeling in self-aware FPGA-based architectures for efficient resource management. In Proceedings of the 10th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC), June 29 to July 1, Bremen, Germany.

Salehie, M. and Tahvildari, L. 2009. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems, 4(14):1–42.

Salvador, R. , Otero, A. , Mora, J. , de la Torre, E. , Riesgo, T. , and Sekanina, L. 2013. Self-reconfigurable evolvable hardware system for adaptive image processing. IEEE Transactions on Computers, 62:1481–1493.

Santambrogio, M.D. 2009. From reconfigurable architectures to self-adaptive autonomic systems. In Proceedings of the International Conference on Computational Science and Engineering (CSE 2009), August 29–31, Vancouver, British Columbia, Canada.

Sohanghpurwala, A.A. , Athanas, P. , Frangieh, T. , and Wood, A. 2011. OpenPR: An open-source partial reconfiguration toolkit for Xilinx FPGAs. In Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW 2011), May 16–20, Anchorage, AK.

Steiner, N. and Athanas, P. 2009. Hardware autonomy and space systems. In Proceedings of the 2009 IEEE Aerospace Conference, March 7–14, Big Sky, MT.

Steiner, N. , Wood, A. , Shojaei, H. , Couch, J. , Athanas, P. , and French, M. 2011. Torc: Towards an open-source tool flow. In Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA ‘11), February 27 to March 1, Monterey, CA.

Ullmann, M. , Hubner, M. , Grimm, B. , and Becker, J. 2004. An FPGA run-time system for dynamical on demand reconfiguration. In Proceedings of the 18th International Parallel and Distributed Processing Symposium, April 26–30, Santa Fe, NM.

Valverde, J. , Rodriguez, A. , Mora, J. , Castañares, C. , Portilla, J. , de la Torre, E. , and Riesgo, T. 2014. A dynamically adaptable image processing application trading off between high performance, consumption and dependability in real time. In Proceedings of the 2014 International Conference on Field Programmable Logic and Applications (FPL 2010), August 31 to September 2, Munich, Germany.

Walder, H. and Platzner, M. 2004. A runtime environment for reconfigurable hardware operating systems. In Proceedings of the 14th International Conference on Field-Programmable Logic and Applications, August 30 to September 1, Leuven, Belgium.

## Industrial Electronics Applications of FPGAs

Aguirre-Dobernack, N. , Guzman-Miranda, H. , and Aguirre, M.A. 2013. Implementation of a machine vision system for real-time traffic sign recognition on FPGA. In Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON’2013), November 10–13, Vienna, Austria.

Barranco, F. , Diaz, J. , Pino, B. , and Ros, E. 2014. Real-time visual saliency architecture for FPGA with top-down attention modulation. IEEE Transactions on Industrial Informatics, 10(3):1726–1735.

Buccella, C. , Cecati, C. , and Latafat, H. 2012. Digital control of power converters—A survey. IEEE Transactions on Industrial Informatics, 8(3):437–447.

Cheng, Ch.-F. , Li, R.-S. , and Chen, J.-R. 2013. Design of the DC leakage current sensor with magnetic modulation-based scheme. In Proceedings 2013 IEEE International Symposium on Industrial Electronics (ISIE’2013), May 28–31, Taipei, Taiwan.

Dagbagi, M. , Hemdani, A. , Idkhajine, L. , Naouar, M.W. , Monmasson, E. , and Slama-Belkhodja, I. 2013. FPGA-based real-time hardware-in-the-loop validation of a 3-phase PWM

rectifier controller. In Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON'2013), November 10–13, Vienna, Austria.
Gomes, L. , Monmasson, E. , Cirstea, M. , and Rodriguez-Andina, J.J. 2013. Industrial electronic control: FPGAs and embedded systems solutions. In Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON'2013), November 10–13, Vienna, Austria.
Gomes, L. and Rodríguez-Andina, J.J. 2013. Guest editorial special section on embedded and reconfigurable systems. IEEE Transactions on Industrial Informatics, 9(3):1588–1590.
Guo, H. , Chen, H. , Xu, F. , Wang, F. , and Lu, G. 2013. Implementation of EKF for vehicle velocities estimation on FPGA. IEEE Transactions on Industrial Electronics, 60(9):3823–3835.
Guzinski, J. and Abu-Rub, H. 2013. Speed sensorless induction motor drive with predictive current controller. IEEE Transactions on Industrial Electronics, 60(2):699–709.
Hace, A. and Franc, M. 2013. FPGA implementation of sliding-mode-control algorithm for scaled bilateral teleoperation. IEEE Transactions on Industrial Informatics, 9(3):1291–1300.
Hasanzadeh, A. , Edrington, C.S. , Stroupe, N. , and Bevis, T. 2014. Real-time emulation of a high-speed microturbine permanent-magnet synchronous generator using multiplatform hardware-in-the-loop realization. IEEE Transactions on Industrial Electronics, 61(6):3109–3118.
Hwang, S.-H. , Liu, X. , Kim, J.-M. , and Li, H. 2013. Distributed digital control of modular-based solid-state transformer using DSP+FPGA. IEEE Transactions on Industrial Electronics, 60(2):670–680.
Jimenez, O. , Lucia, O. , Urriza, I. , Barragan, L.A. , and Navarro, D. 2014. Analysis and implementation of FPGA-based online parametric identification algorithms for resonant power converters. IEEE Transactions on Industrial Informatics, 10(2):1144–1153.
Juarez-Abad, J.A. , Linares-Flores, J. , Guzmán-Ramirez, E. , and Sira-Ramirez, H. 2014. Generalized proportional integral tracking controller for a single-phase multilevel cascade inverter: An FPGA implementation. IEEE Transactions on Industrial Informatics, 10(1):256–266.
Kobravi, K. , Iravani, R. , and Kojori, H.A. 2013. Three-leg/four-leg matrix converter generalized modulation strategy—Part II: Implementation and verification. IEEE Transactions on Industrial Electronics, 60(3):860–872.
Liu, J. and Dinavahi, V. 2014. A real-time nonlinear hysteretic power transformer transient model on FPGA. IEEE Transactions on Industrial Electronics, 61(7):3587–3597.
Lu, X. , Chen, H. , Gao, B. , Zhang, Z. , and Jin, W. 2015. Data-driven predictive gearshift control for dual-clutch transmissions and FPGA implementation. IEEE Transactions on Industrial Electronics, 62(1):599–610.
Lu, Z.-G. , Zhao, L.-L. , Zhu, W.-P. , Wu, C.-J. , and Qin, Y.-S. 2013. Research on cascaded three-phase-bridge multilevel converter based on CPS-PWM. IET Power Electronics, 6(6):1088–1099.
May, K. and Krougjicof, N.K. 2013. Moving target detection for sense and avoid using regional phase correlation. In Proceedings 2013 IEEE International Conference on Robotics and Automation (ICRA'2013), May 6–10, Karlsruhe, Germany.
Miura, Y. , Inubushi, K. , Ito, M. , and Ise, T. 2014. Multilevel modular matrix converter for high voltage applications. Control, design and experimental characteristics. In Proceedings of the 40th Annual Conference of the IEEE Industrial Electronics Society (IECON'2014), October 30 to November 1, Dallas, TX.
Monmasson, E. and Cirstea, M. 2013. Guest editorial special section on industrial control applications of FPGAs. IEEE Transactions on Industrial Informatics, 9(3):1250–1252.
Monmasson, E. and Cirstea, M.N. 2007. FPGA design methodology for industrial control systems—A review. IEEE Transactions on Industrial Electronics, 54(4):1824–1842.
Monmasson, E. , Idkhajine, L. , Cirstea, M.N. , Bahri, I. , Tisan, A. , and Naouar, M.W. 2011a. FPGAs in industrial control applications. IEEE Transactions on Industrial Informatics, 7(2):224–243.
Monmasson, E. , Idkhajine, L. , and Naouar, M.W. 2011b. FPGA-based controllers. IEEE Industrial Electronics Magazine, 5(1):14–26.
Morales-Caporal, R. , Bonilla-Huerta, E. , Hernández, C. , Arjona, M.A. , and Pacas, M. 2013. Transducerless acquisition of the rotor position for predictive torque controlled PM synchronous machines based on a DSP-FPGA digital system. IEEE Transactions on Industrial Informatics, 9(2):799–807.
Naouar, M.-W. , Monmasson, E. , Naassani, A.A. , Slama-Belkhodja, I. , and Patin, N. 2007. FPGA-based current controllers for AC machine drives—A review. IEEE Transactions on

Industrial Electronics, 54(4):1907–1925.
Nikolic, J. , Rehder, J. , Burri, M. , Gohl, P. , Leutenegger, S. , Furgale, P.T. , and Siegwart, R. 2014. A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In Proceedings 2014 IEEE International Conference on Robotics and Automation (ICRA'2014), May 31 to June 7, Hong Kong, China.
Phuong, T.T. , Ohishi, K. , Yokokura, Y. , and Mitsantisuk, C. 2014. FPGA-based high-performance force control system with friction-free and noise-free force observation. IEEE Transactions on Industrial Electronics, 61(2):994–1008.
Pierce, B. and Cheng, G. 2014. Versatile modular electronics for rapid design and development of humanoid robotic subsystems. In Proceedings 2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM'2014), July 8–11, Besancon, France.
Prabhala, V.A. , Cespedes, M. , and Sun, J. 2012. Implementation of DQ domain control in DSP and FPGA. In Proceedings of the 2012 Twenty-Seventh Annual IEEE Applied Power Electronics Conference and Exposition (APEC'2012), February 5–9, Orlando, FL.
Rodriguez-Andina, J.J. , Moure, M.J. , and Valdes, M.D. 2007. Features, design tools, and application domains of FPGAs. IEEE Transactions on Industrial Electronics, 54(4):1810–1823.
Rodriguez-Andina, J.J. , Valdes, M.D. , and Moure, M.J. 2015. Advanced features and industrial applications of FPGAs—A review. IEEE Transactions on Industrial Informatics, 11(4):853–864.
Rodriguez-Araujo, J. , Rodriguez-Andina, J.J. , Farina, J. , and Chow, M.-Y. 2014. Field-programmable System-on-Chip for localization of UGVs in an indoor iSpace. IEEE Transactions on Industrial Informatics, 10(2):1033–1043.
Rodriguez-Araujo, J. , Rodriguez-Andina, J.J. , Farina, J. , Vidal, F. , Mato, J.L. , and Montealegre, M.A. 2012. Industrial laser cladding systems: FPGA-based adaptive control. IEEE Industrial Electronics Magazine, 6(4):35–46.
Schmid, K. and Hirschmuller, H. 2013. Stereo vision and IMU based real-time ego-motion and depth image computation on a handheld device. In Proceedings 2013 IEEE International Conference on Robotics and Automation (ICRA'2013), May 6–10, Karlsruhe, Germany.
Schmitt, A. , Richter, J. , Jurkewitz, U. , and Braun, M. 2014. FPGA-based real-time simulation of nonlinear permanent magnet synchronous machines for power hardware-in-the-loop emulation systems. In Proceedings of the 40th Annual Conference of the IEEE Industrial Electronics Society (IECON'2014), October 30 to November 1, Dallas, TX.
Senicar, F. , Dopker, M. , Bartsch, A. , Kruger, B. , and Soter, S. 2014. Inverter based method for measurement of PMSM machine parameters based on the elimination of power stage characteristics. In Proceedings of the 40th Annual Conference of the IEEE Industrial Electronics Society (IECON'2014), October 30 to November 1, Dallas, TX.
Sepulveda, C.A. , Munoz, J.A. , Espinoza, J.R. , Figueroa, M.E. , and Baier, C.R. 2013. FPGA v/s DSP performance comparison for a VSC-based STATCOM control application. IEEE Transactions on Industrial Informatics, 9(3):1351–1360.
Shahbazi, M. , Poure, P. , Saadate, S. , and Zolghadri, M.R. 2013. FPGA-based reconfigurable control for fault-tolerant back-to-back converter without redundancy. IEEE Transactions on Industrial Electronics, 60(8):3360–3371.
Smidl, V. , Nevdev, R. , Kosan, T. , and Peroutka, Z. 2013. FPGA implementation of marginalized particle filter for sensorless control of PMSM drives. In Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON'2013), November 10–13, Vienna, Austria.
Wang, C. , Li, W. , and Belanger, J. 2013. Real-time and faster-than-real-time simulation of modular multilevel converters using standard multi-core CPU and FPGA chips. In Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON'2013), November 10–13, Vienna, Austria.
Wang, Y. , Yang, K. , He, C. , and Chen, G. 2014. A harmonic elimination approach based on moving average filter for cascaded DSTATCOM. In Proceedings of the 40th Annual Conference of the IEEE Industrial Electronics Society (IECON'2014), October 30 to November 1, Dallas, TX.
Wen, H. , Cheng, D. , Teng, Z. , Guo, S. , and Li, F. 2014. Approximate algorithm for fast calculating voltage unbalance factor of three-phase power system. IEEE Transactions on Industrial Informatics, 10(3):1799–1805.
Xia, G.-M. , Qiu, A.-P. , Shi, Q. , and Yan, S. 2013. Test and evaluation of a silicon resonant accelerometer implemented in SOI technology. In Proceedings 2013 IEEE Sensors, November 3–6, Baltimore, MD.

Xu, Q. 2014. Design and smooth position/force switching control of a miniature gripper for automated microhandling. IEEE Transactions on Industrial Informatics, 10(2):1023–1032.
Xu, Q. 2015. Robust impedance control of a compliant microgripper for high-speed position/force regulation. IEEE Transactions on Industrial Electronics, 62(2):1201–1209.
Zhu, W. , Lamarche, T. , Dupuis, E. , Jameux, D. , Barnard, P. , and Liu, G. 2013. Precision control of modular robot manipulators: The VDC approach with embedded FPGA. IEEE Transactions on Robotics, 29(5):1162–1179.
Zhu, Z. , Li, X. , Rao, H. , Wang, W. , and Li, W. 2014. Testing a complete control and protection system for multi-terminal MMC HVDC links using hardware-in-the-loop simulation. In Proceedings of the 40th Annual Conference of the IEEE Industrial Electronics Society (IECON'2014), October 30 to November 1, Dallas, TX.