CHAPTER

Introduction 1

1.1 TRENDS IN SYSTEM-ON-CHIP DESIGN

Advances in silicon technology continue to leapfrog projections: At the time this book was being written, we were seeing numerous announcements for billion-transistor chips, whereas only a few years ago integrated circuit transistor counts were in the millions. Such single chip integrated circuits are commonly referred to as *system-on-chip* (SoC), and typically consist of several complex heterogeneous components such as programmable processors, dedicated (custom) hardware to perform specific tasks, on-chip memories, input–output interfaces, and an on-chip communication architecture that serves as the interconnection fabric for communication between these components. The dual forces of advances in technology, coupled with an insatiable demand for convergent computing devices (e.g., smart phones that include cameras, GPS devices, MP3 players) have fueled the need for complex chips that incorporate multiple processors dedicated for specific computational needs. These emerging *multiprocessor system-on-chip* (MPSoC) designs typically consist of multiple microprocessors, and tens to hundreds of additional components. Figure 1.1(a) shows an example of a small MPSoC from the multimedia domain, that incorporates two ARM9 microprocessors running embedded software, several on-chip memories, DMA (direct memory access) and LCD controllers, peripherals (e.g., timer and interrupt controller), and external interfaces (e.g., USB and Ethernet), all of which are integrated via an on-chip bus architecture consisting of multiple shared interconnected buses. Another example of a more complex MPSoC is the IBM Cell [1] (shown in Fig. 1.1(b)) used in the Sony PlayStation 3 gaming console. It consists of nine processors—eight special-purpose synergistic processing units (SPU) that perform dedicated computing tasks, and a single general purpose power processor unit that performs generalized processing, and oversees the activities on the chip. Additionally, the Cell has on-chip Level 2 (L2) cache memory, interface components to interact with external electronic systems, and a ring bus-based on-chip communication architecture that facilitates data communication between the components on the chip. Figure 1.1 clearly shows that emerging MPSoCs will use a variety of on-chip bus communication architectures that are tuned to the requirements of the application, architecture, as well as the available technology.

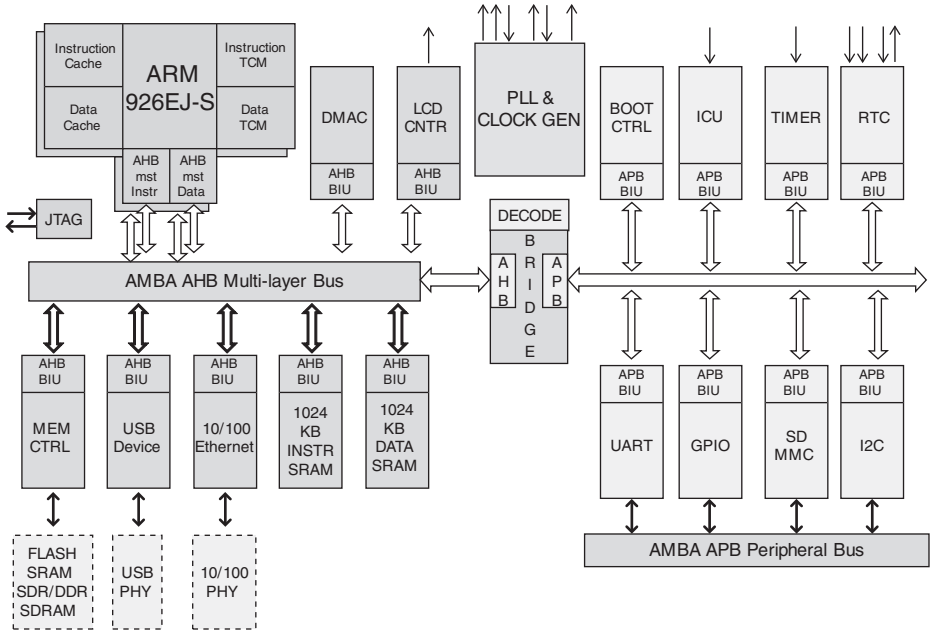

FIGURE 1.1

(a) A multimedia SoC employing a bus topology

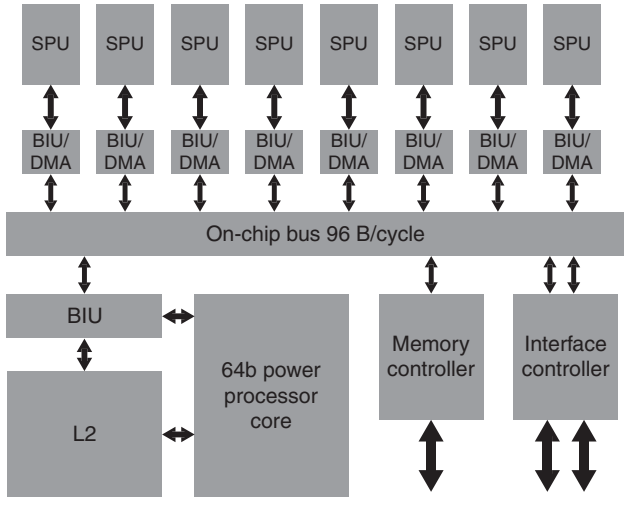

FIGURE 1.1

(b) the IBM Cell processor employing a ring topology

The availability of such a large number of devices on a chip enables new ways of realizing system functionality using a combination of software and hardware, where each MPSoC computational element can be viewed as an *intellectual property* (IP) block. These IP blocks can be a library element from a previous design

(i.e., a *hard macro*), a parameterizable component to meet design requirements (i.e., a *soft macro*), or a custom-designed computational block to meet the performance, power, and other design constraints. The architectural landscape for these computational IP blocks range from off-the-shelf *general purpose processor* (GPP) cores, through *application-specific instruction-set processors* (ASIPs), *reconfigurable cores* (RCs), and custom-designed *application-specific integrated circuit* (ASIC) blocks. In going from a GPP to an ASIC block, the system designer is typically trading off flexibility (general purpose programmability) vs. performance, power efficiency, and cost. Furthermore, a plethora of domain-specific programmable processor cores have been developed—either as standalone computational engines or as computational assists to address the specific needs of an application (e.g., *digital signal processors* (DSPs), network processors, and *graphics processing units* (GPUs)).

In the context of such IP-based design methodologies, the communication architecture is both a key enabler, as well as a key differentiator for realizing complex MPSoCs. As will be described in Section 1.3, contemporary IP-based design flows are based on the premise of design reuse, with the system designer stitching together disparate IP blocks using the critical communication architecture fabric for data communication between the IP blocks. Thus the communication architecture *enables* mixing and matching of different IP blocks to create an MPSoC. Furthermore, the communication architecture provides designers the ability to explore multiple communication schemes, topologies and protocols, thereby providing product *differentiation* to meet a diverse, multi-dimensional set of design constraints, including performance, power/energy/temperature, cost, reliability, and time-to-market.

1.2 COPING WITH SoC DESIGN COMPLEXITY

The continuing improvements in silicon technology provide a great opportunity (we can integrate more and more devices on an integrated circuit), but also result in the famous "designer productivity gap," as shown in Fig. 1.2. In this figure, the *x*-axis shows progression in time, while the *y*-axis has two lines: The solid line shows the rate of growth of chip complexity (as measured by the number of logic transistors per chip), while the dotted line captures designer productivity (as measured by the average number of transistors designed by a staff engineer in a month). As can be seen in Fig. 1.2, the relentless march of progress in silicon technology is far outstripping our ability to effectively utilize these transistors for working designs in a short amount of time. As product lifetimes continue to shrink, time-to-market becomes a key design constraint.

Several strategies have been developed to cope with this widening designer productivity gap. We outline two major strategies in this chapter. The first is the elevation of the design process for architects to the *electronic system level* (ESL), where early design decisions, exploration and platform decisions are made above the traditional logic/register transfer level (RTL) (this is described further in Section 1.3). The second is aggressive exploitation of *design reuse* at the ESL. The reuse factor can be enhanced by increasing the granularity of library elements

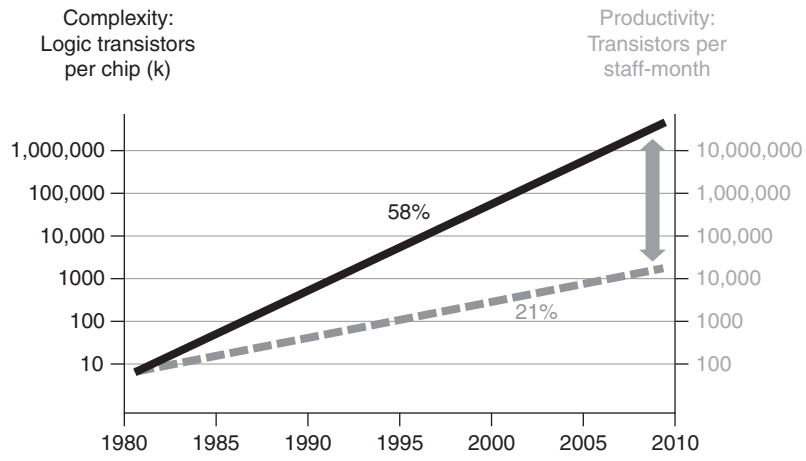

FIGURE 1.2

Designer productivity gap

from standard cells to hardware (e.g., custom accelerator) and software (e.g., processor core) IP blocks. Collections of these IP blocks can be configured into *platforms* designed for specific application domains, which further increases design reuse. Furthermore, the move toward migration of functionality to software allows reuse of software libraries that can be mapped onto existing platforms and IP blocks, which again enhances design reuse. The term *Platform-based Design* [2] is often used to represent a preconfigured silicon architecture using a number of software and hardware IPs, and is aimed at a specific application domain or even an end application (e.g., video compression, network packet processing). Such platforms typically allow for some degree of customization through architectural parameters for each IP, and often include an accompanying software toolchain to ease application development. The system architect can tweak platform parameters to tune the design for a range of performance, power, and *quality of service* (QoS) attributes.

In this regard, it is important to note that the communication architecture may be customized as well. Indeed, every instance of a silicon platform uses a specific class of communication architecture, and the platform provider may allow the architect to specify or modify some key communication parameters (e.g., bus widths and protocols) to allow for further customization of the design. As we will emphasize throughout this book, the on-chip communication architecture fabric in an MPSoC platform has a significant impact on its power, performance, cost, reliability, and time-to-market.

1.3 ESL DESIGN FLOW

Figure 1.3 shows an idealized MPSoC ESL design flow. Most designs start from customer or marketing requirement specifications that describe the overall application

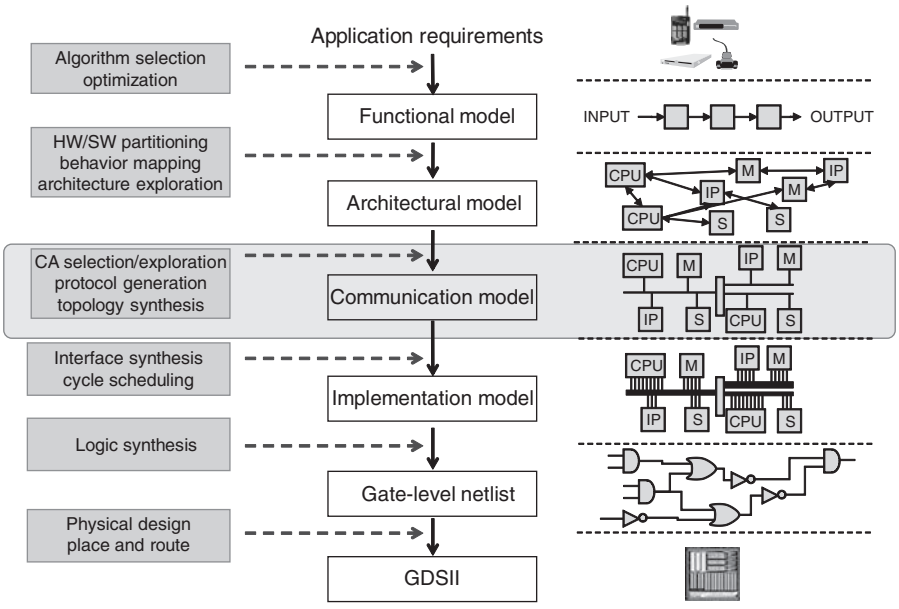

FIGURE 1.3

Ideal ESL design flow for MPSoCs (the on-chip communication architecture design phase is highlighted)

at a very abstract level. In the first step, designers select the algorithms to use, perform optimizations, and create a *functional model* of the application in a high level language such as C/C++. This is a very high level sequential-execution model of the application which captures its entire functionality. In the next step, designers perform hardware/software partitioning—mapping part of the functionality into hardware components and the remaining functionality to software. Architecture exploration is performed to select the best possible hardware or software candidates for the functional (or behavior) mapping. The result is an *architecture model* of the system where the computation entities in the system have been defined. Note, however, that these entities communicate with each other in an abstract manner, for instance using high level message passing. It is the responsibility of the subsequent *on-chip communication architecture synthesis* step to define a communication architecture for the system. This synthesis step performs design space exploration to select a particular communication architecture (e.g., hierarchical bus, ring bus, bus matrix) that best satisfies the constraints of the application. Once a communication architecture has been selected, further exploration is performed to define its topology and values for protocol parameters (e.g., arbitration scheme, bus width, and frequency). The resulting model is called a *communication model*. It is an enhanced architecture model of the system with well-defined computation and communication entities. The model is further refined in the next step, in which behaviors inside computation blocks are scheduled at cycle boundaries (cycle-accurate behavior). The interface between

the computation blocks and the communication architecture is also refined to a pin-accurate level, with all the signals needed for communication being explicitly modeled. These steps lead to the creation of a detailed *implementation model*. This model is essentially an RTL model, which can be an input to standard logic synthesis tools to create a gate level netlist of the design. Subsequently, the designer performs placement of the various modules on the chip floor plan, followed by a routing step to connect the modules together. The resulting GDSII file is then handed to a semiconductor foundry for fabricating the MPSoC design.

A major portion of this book is devoted to the communication architecture phase in the MPSoC ESL design flow, as highlighted in Fig. 1.3. Note that this MPSoC ESL flow is a high level illustration of an actual flow, and thus does not show many of the iteration and verification steps that must be performed concurrently with the design flow. In addition, while the MPSoC design flow consists of the basic steps as shown in Fig. 1.3, in practice designers often merge some of these steps, or split one or more of the steps for a more detailed treatment of the problem.

1.4 ON-CHIP COMMUNICATION ARCHITECTURES: A QUICK LOOK

The components in an MPSoC design invariably need to communicate with each other during application execution. For instance, a microprocessor fetches instructions from memory components, or writes to external memories by sending data to an on-chip memory controller. It is the responsibility of the on-chip communication architecture to ensure that the multiple, co-existing data streams on the chip are correctly and reliably routed from the source components to their intended destinations. In addition to correctness, the on-chip communication architecture must provide latency or bandwidth guarantees to ensure that the application performance constraints are satisfied. A latency guarantee implies that a data unit must traverse the communication architecture and reach its destination within a finite amount of time, determined by a latency bound (e.g., 40ns from source to destination). A bandwidth guarantee implies that a group of data units must traverse a portion of the communication architecture at a certain data rate, as determined by the bandwidth requirements (e.g., 100 megabits/second from source to destination). Depending on the performance requirements of an application, various types of on-chip communication architectures can be used, as described in following subsection.

1.4.1 Types of On-Chip Communication Architectures

A basic building block of most on-chip communication architectures in MPSoC designs is the single shared bus. This is the simplest on-chip communication architecture, consisting of a set of shared, parallel wires to which various components are connected. Only one component on the bus can have control of the shared

wires at any given time to perform data transfers. This limits the parallelism and achievable performance in the system, which makes it unsuitable for most MPSoC applications that can have tens to hundreds of components. Consequently, the single shared bus architecture is not scalable to meet the demands of MPSoC applications.

Figure 1.4 shows various kinds of on-chip communication architectures that are used in MPSoC designs. Many contemporary MPSoC designs mostly use shared bus-based communication architectures. Figure 1.4(a) shows a hierarchical shared bus architecture, which consists of a hierarchy of buses interconnected using bridge components. Shared buses higher up in the hierarchy are typically operated at higher clock frequencies, and are used to connect high speed, high performance components. On the other hand, shared buses lower down in the hierarchy are operated at lower frequencies to save power, and connect high latency, low performance components. Figure 1.4(b) shows a ring type bus, similar to that used in the IBM Cell MPSoC. The ring bus is actually a set of unidirectional, concentric and pipelined buses which allow high frequency operation and high bandwidth transfers between components on the bus. Figure 1.4(c) shows an ad-hoc bus architecture, where buses are operated at different frequencies and components can have point-to-point links with each other, as needed. Finally, Figure 1.4(d) shows the bus matrix (or crossbar bus) where a crossbar type architecture connects processors (and their local bus components) on the left to memories and peripherals on the right. This kind of architecture is a combination of shared bus and point-to-point interconnections.

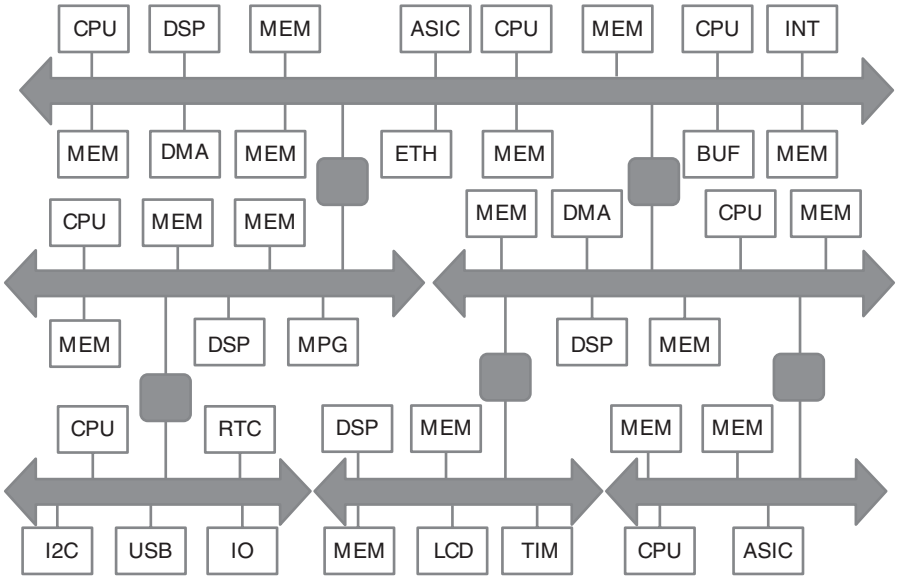

FIGURE 1.4

MPSoC bus-based on-chip communication architectures: (a) hierarchical bus

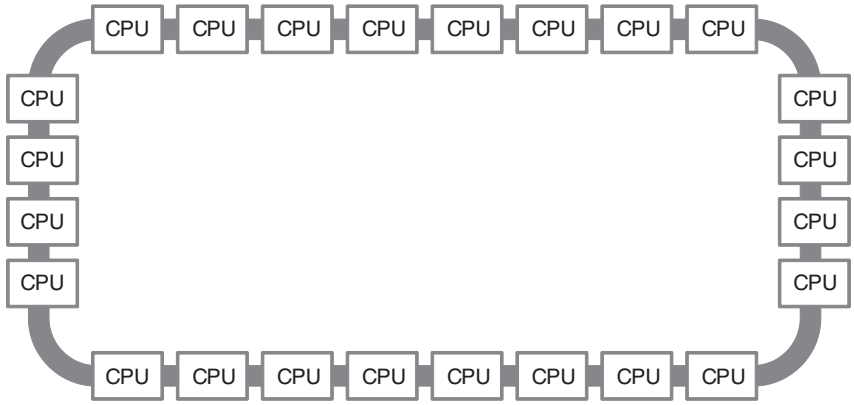

FIGURE 1.4

(b) Ring bus

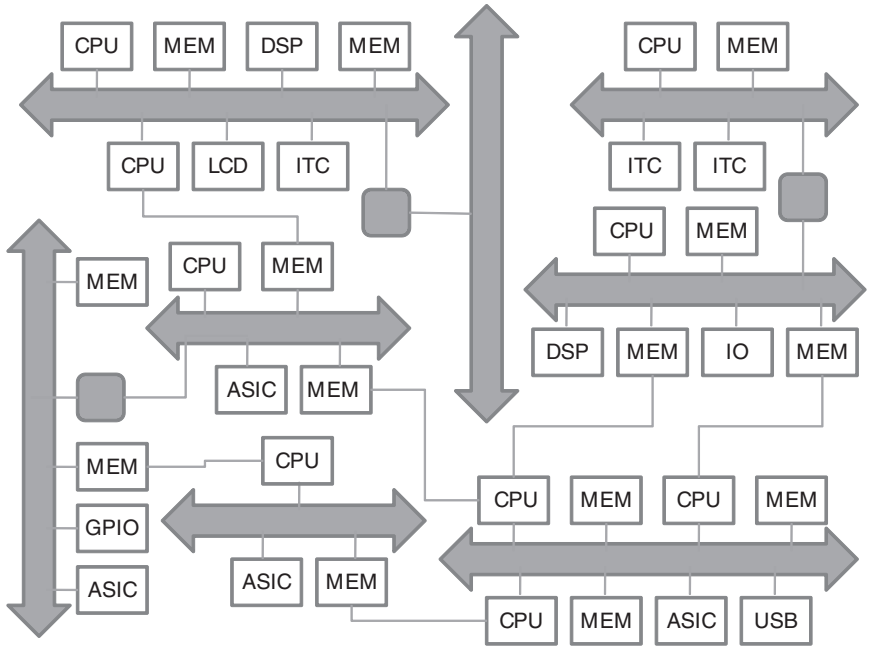

FIGURE 1.4

(c) Ad-hoc bus

Each of the above bus-based on-chip communication architectures is defined by its two major constituents: *topology* and *protocol parameters*. The topology of a communication architecture refers to how the buses are interconnected together, and how the various components are mapped to each of the buses. The protocol parameters refer to such parameters as arbitration schemes, bus widths, bus clock frequencies, buffer sizes, and burst transfer sizes, which are specific to

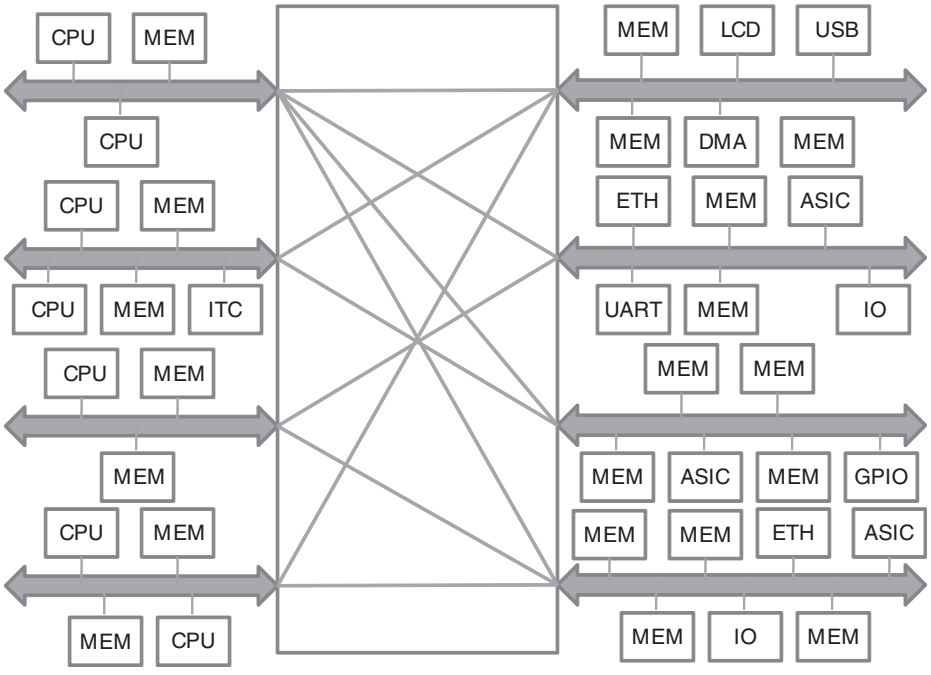

FIGURE 1.4

(d) bus matrix (or crossbar bus)

the protocol used by the communication architecture. Designing a communication architecture thus implies determining both its topology and protocol parameter values.

It has been projected that bus-based architectures cannot scale up with an increasing number of components, and also given the increasing amount of on-chip wire delays (resulting in timing unpredictability).Thus future MPSoC designs with hundreds of components will make use of network-on-chip (NoC) communication fabrics, where instead of shared buses, packet switched network fabrics with routers are used to transfer data between on-chip components. However, NoCs are still in their early phase of research and development, and concrete implementations of NoC-based MPSoCs are only now beginning to appear. The primary focus of this book is for bus-based on-chip communication architectures, although we will also introduce NoCs toward the end of this book.

Having introduced the different types of communication architectures used in MPSoC designs, we now look at how increasing application complexity and technology scaling have affected on-chip communication in recent years.

1.4.2 Impact of Increasing Application Complexity

With increasing application complexity and the rising number and variety of components being integrated into MPSoC designs, communication between on-chip

components is playing an increasingly critical role in ensuring that application performance constraints are satisfied. Due to the increasing interdependence of various components on a chip, a seemingly insignificant bottleneck in transferring a single data stream between two components can stall the entire chip, leading to a functional failure of the chip. This not so uncommon scenario is a consequence of the sheer number of simultaneous data streams traversing a typical MPSoC chip at any given time, and being managed by finite communication resources (wires, buffers). Figure 1.5 shows the rising performance requirements of emerging applications, which will inevitably increase the amount of data communication traffic on a chip and further increase the probability of unforeseen bottlenecks encountered in on-chip communication in the future.

To cope with the increasing MPSoC performance requirements, on-chip communication architectures have also undergone an evolution in complexity—from shared buses, to hierarchical shared buses, and onto bus matrix (or crossbar bus) architectures. This has had two notable consequences for on-chip communication architecture design. Firstly, since advanced bus-based architectures such as the bus matrix make use of many more wires and logic components to support high performance requirements, they have a much larger power consumption and area overhead. Thus design decisions made during communication architecture selection and implementation must take into account not only the supported performance, but also ensure that overall chip power and area constraints are not violated by the communication architecture. Secondly, these advanced communication architectures have enormous design spaces that are not so easy to explore. The combination of different topologies, component mapping choices, and protocol parameter values for a communication architecture can easily create a design space with billions of possible configurations, which makes it incredibly difficult for designers to choose which configurations to explore in the finite amount of time available in an MPSoC design cycle. Thus the task of designing on-chip

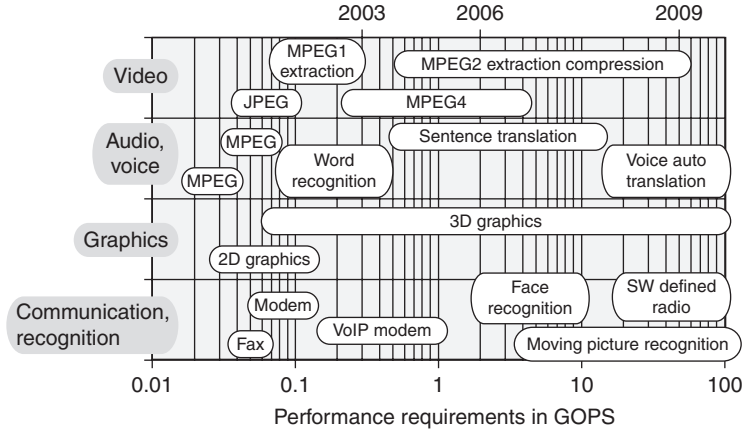

FIGURE 1.5

Increasing performance requirements for emerging applications [3,4]

communication architectures today has become a major challenge for designers, requiring a careful, time-consuming decision process to adequately balance the different constraints (power, performance, area) of the application.

1.4.3 Impact of Technology Scaling

The increasing levels of component integration would not have been possible without technology scaling that has enabled a reduction in transistor size, allowing more of them to be integrated into the same area with each passing technology generation. However, these technological advances that have ushered the industry into the *deep submicron* (DSM) era, with the ongoing commercialization of the 90 and 65nm processes, have introduced new challenges for on-chip communication architecture design. Precise control of the fabrication process in DSM technologies is almost impossible, leading to process uncertainties that cause nonuniformity of sheet resistance and an increase in coupling noise between adjacent wires in buses. In addition, decreasing wire pitch and increasing aspect ratio with technological advances further accelerates these issues. The end result of these factors is that signal propagation delay on wires (i.e., interconnect delay) is increasing with each technology generation, which puts a limit on the communication performance.

According to the *International Technology Roadmap for Semiconductors* (ITRS) 2005 predictions (Fig. 1.6) [4], the gap between interconnection delay and gate delay will increase to 9:1 at the 65nm technology. This is in sharp contrast to the 2:1 gap between interconnection delay and gate delay at the 180nm technology.

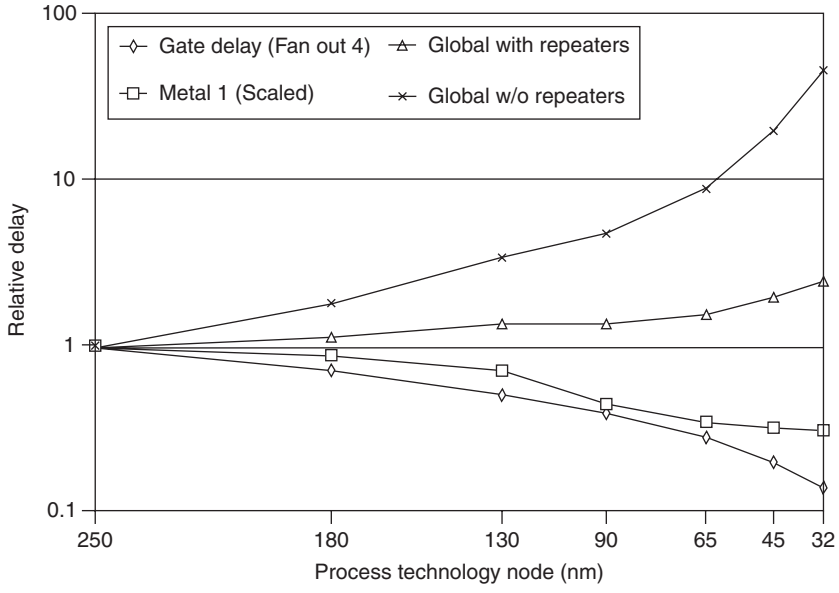

FIGURE 1.6

Relative delay comparison of wires vs. process technology [4]

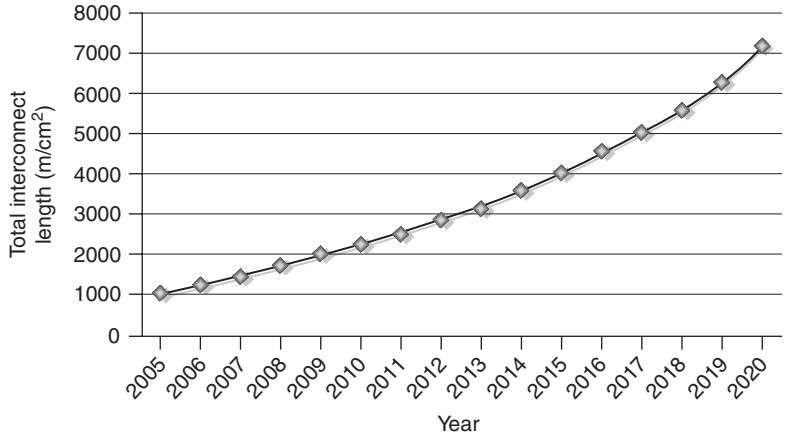

FIGURE 1.7

Trend of total interconnect length on a chip [4]

This indicates that communication, and not computation, will be the key performance bottleneck in DSM technologies. In addition, total wire length on a chip is expected to amount to 2.22km/cm^2 by the year 2010 (Fig. 1.7) [4]. Another observation is the increase of power dissipation due to the charging and discharging of interconnection wires on a chip. According to [4,5], the capacitance portion of the interconnect contributes to about 30% of the total capacitance of a chip, and soon the interconnect will consume about 50 times more power than logic circuits [6]. This means that for MPSoC designs in the DSM era, the performance, power consumption, cost, and area will be much more influenced by the on-chip communication architecture than the gates on the chip. Thus moving forward, MPSoC designs will necessarily have to be interconnect-aware and address communication architecture issues very early in the design process.

1.5 BOOK OUTLINE

This book attempts to provide a comprehensive overview of various aspects of on-chip communication in MPSoCs, and gives insight into why on-chip communication architectures are becoming a critical issue in MPSoC designs. The next chapter presents basic concepts of bus-based communication architectures—introducing commonly used terminology, structural components, wiring issues, and DSM effects associated with bus architectures. Chapter 3 gives an overview of some of the prevailing standards in on-chip communication architectures (such as AMBA 2.0/3.0, Altera Avalon, IBM CoreConnect, Sonics SMART Interconnect, STMicroelectronics STBus, and Opencores Wishbone) that are used to facilitate component integration in MPSoC designs. While these bus-based communication architecture standards define the interface between components and the bus architecture, as well as the bus architecture that implements the data transfer protocol, socket-based standards give a lot more freedom to a designer, with respect to the choice and implementation of the bus architecture, since they only

provide definitions for designing component interfaces. In principle, such socket-based interface standards enable designers to reuse IPs designed for disparate communication protocols, thereby facilitating an IP-based design methodology. An overview of some of the popular socket-based interface standards (such as open core protocol (OCP), virtual component interface (VCI), and device transaction level (DTL)) is also presented in the chapter.

Having presented a background on the basic concepts of bus-based communication architectures, and prevailing standards, the next few chapters address the important problem of understanding the on-chip communication architecture design space, to aid in selecting the best communication architecture configuration for an application. Chapter 4 looks at models for the performance estimation of communication architectures. Various stochastic, simulation, and hybrid modeling abstractions are described here, that trade-off modeling complexity with estimation speed. These models capture details of the communication architecture design space and allow designers to estimate the performance of different communication architecture configurations. Chapter 5 presents various models for power and thermal estimation of communication architectures that allow systems designers to perform early estimation of the power and thermal characteristics for different communication architecture configurations. The chapter also highlights the need for PVT (process, voltage, temperature) variation-aware power estimation in future ultra DSM (UDSM) technologies. These performance and power/thermal models for communication architectures are used as part of various techniques, presented in Chapter 6, to select, configure, and design communication architectures that meet the requirements of a given application. While the emphasis here is on high level exploration based synthesis of communication architectures, novel approaches that couple physical implementation-awareness during early exploration, and co-synthesize memory and communication architectures are also presented, along with an overview of lower level physical and circuit level techniques for communication architecture design.

Once the various estimation models and design approaches have been studied, the next few chapters focus on specific research efforts in the area of communication architecture design. Chapter 7 gives an overview of a large body of work on various encoding techniques used to transform the data to be transmitted on communication architectures, in order to reduce switching power, propagation delay, crosstalk noise (both inductive and capacitive), and transmission errors. Schemes that jointly optimize for several design goals, and provide trade-offs to the designer between power, performance, and reliability, are also presented. Chapter 8 looks at custom bus architectures (such as split/segmented, serial, CDMA, asynchronous, and reconfigurable architectures) that attempt to address the shortcomings of standard on-chip communication architectures by utilizing new topologies and protocols to obtain improvements for common design goals, such as performance and power. Chapter 9 takes a step back to present various refinement methodologies that aim to take the communication architecture design process from the functional level down to the implementation level (Fig. 1.3) using a combination of novel modeling abstractions and refinement techniques. The chapter then dives into the problem of interface synthesis, which is a critical issue in such methodologies. Techniques are presented for efficiently

synthesizing interfaces to correct protocol mismatches, and to ensure that interfaces using the same protocol but having different level of details correctly interact with each other. Chapter 10 tackles verification and security issues in communication architecture design. The chapter first presents techniques to verify the properties and constraints of on-chip communication protocols and communication architecture logic components, such as arbiters, and techniques for the verification of IP blocks being integrated into an SoC, to ensure that they are compliant with the on-chip communication protocol. Subsequently, the focus shifts to security issues, with a look at security features built into bus-based communication architecture standards, and an overview of research efforts that modify existing communication architecture standards to enhance overall system security.

The final part of the book presents trends in the design of on-chip communication architectures over the next several years. Chapter 11, contributed by Savidis and Friedman, presents physical design trends for on-chip communication architecture design. The focus here is on understanding the effect of DSM on interconnect design, as well as to give an overview of low power, high speed circuit design techniques that can be used in global power, and clock distribution networks. A discussion on emerging 3-D interconnects is also presented. Chapter 12 outlines networks-on-chip (NoCs), a promising new direction in on-chip communication design, and a hot area of research at the time of writing this book. The chapter describes various design aspects that characterize NoCs, such as the topology, switching schemes, routing algorithms, flow control mechanisms, clocking strategies, and QoS support. A survey of several NoC architectures that have been proposed in literature is also presented, along with a discussion on the current status and open problems in the NoC domain. Chapter 13 concludes the book by presenting three emerging technologies on the horizon that propose new paradigms for replacing traditional on-chip metallic interconnects. The chapter presents an overview and challenges associated with *optical interconnects*, that make use of light and an on-chip optical medium to transfer data; *RF/wireless interconnects*, that transfer data on a chip wirelessly using transmitting and receiving antennas integrated on the same chip; and *carbon nanotubes*, which have been shown to be much less susceptible to DSM effects and have been shown to have lower power dissipation and better performance than metallic interconnects. All three of these emerging technologies have several issues and open problems (such as the need for improvements in fabrication technology) that must be resolved before they can be adopted as part of on-chip interconnect fabrics. With the rapid advances in technology, however, it is only a matter of time before one or more of these technologies becomes feasible and advantageous to use in tomorrow's ultra large scale integrated (ULSI) designs.

REFERENCES

[1] IBM Cell Project, http://www.research.ibm.com/cell.

[2] K. Keutzer, S. Malik, A. Newton, J. Rabaey and A. Sangiovanni-Vincentelli, "System level design: Orthogonolization of concerns and platform-based design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 12, 2000, pp. 1523–1543.

[3] N. Ventroux, F. Blanc, R. David and T. Collette, "Reconfigurable multiprocessor system-on-chip for embedded applications," *6th International Forum on Application-Specific Multi-Processor SoC*, Colorado, USA, 2006.

[4] Semiconductor Industry Association. The International Technology Roadmap for Semiconductors, 2005 edition. SEMATECH: Austin, TX., 2005.

[5] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, Norwell, MA, 1995.

[6] W. J. Dally, "Computer architecture is all about interconnect," in *8th International Symposium on High-Performance Computer Architecture*, Cambridge, MA, 2002.