Tutorial 2 System-on-Chip Design

Grant Martin and Henry Chang
Cadence Design Systems
555 River Oaks Parkway
San Jose, CA 95134, USA
gmartin@cadence.com;henry@cadence.com

Abstract

This tutorial is a general introduction to System-on-Chip (SoC) design. In the paper, we will discuss four areas: first, an overview of SoC including descriptions of the major approaches and motivating factors behind this development. Next, we will briefly summarise key design methodologies, processes and flows. SoC, by its nature, involves embedded software (ESW) in many designs, so we will summarise the key areas and requirements for effective ESW development. Finally, we will briefly describe some of the next-generation, advanced concepts that are emerging for SoCs. To maximise the usefulness of this paper, an extensive set of references is given.

Introduction

Before discussing approaches to SoC design, let's define an SoC. We use the definition from [1], to wit: "...SOC design is defined as a complex IC that integrates the major functional elements of a ***complete end-product*** into a single chip or ***chipset***. In general, SOC design incorporates a ***programmable processor***, on-chip memory, and accelerating function units implemented in hardware. It also interfaces to ***peripheral*** devices and/or the real world. SOC designs encompass both hardware and ***software*** components. Because SOC designs can interface to the ***real world***, they often incorporate ***analog*** components, and can, in the future, also include opto/microelectronic mechanical system (***O/MEMS***) components."

We can update this definition in three key areas: more than ever, SoC designs feature increasing amounts of embedded SW; in addition, SoC implementation fabrics incorporate growing amounts of reconfigurable logic, and finally, the custom and analogue IC world is adding more digital design to their devices, leading to more mixed-signal SoCs. Regarding reconfigurable approaches, traditional FPGA providers are now offering what they call "Platform FPGAs", incorporating custom-design fixed programmable processors cores, together with hundreds of thousands or millions of gates of reconfigurable logic. Moving from the other direction, traditional SoC and application-specific integrated circuit (ASIC) providers are beginning to offer reconfigurable logic regions as part of their SoC parts [2].

What motivates the move to SoC? The main motivation is the advance in silicon process technology that increasingly allows a complete system to be designed into one or a few integrated devices. Once this began to be possible, it was irresistible for designers and product managers to take advantage of the capability to map their products into highly integrated chips, taking advantage of space and power reductions, and increased performance. Although there are limits to the feasibility of integrating multiple types of design into one device [3], for digital and mixed-signal designs the advance to SoC has been inexorable.

Fundamental to the possibility of highly integrated SoC designs is the notion of *design reuse*.

Overview of System-On-Chip Design

Chip designs have for the last 20 years reused design elements. What has been changing has been the level of reuse abstraction. In an ASIC style flow, involving RTL logic synthesis and automated standard cell place and route, the reuse abstraction has been at the basic cell level, where a cell represents a few gates of complexity. In addition, reuse of standard modules produced by generators or by hand, such as memories etc. has been common.

SoC design has involved the reuse of more complex elements at higher levels of abstraction. Block-based design, which involves partitioning, designing and assembling SoCs using a hierarchical block-based approach, has used the Intellectual Property (IP) block as the basic reusable element. This might be an interface function such as a PCI or 1394 bus interface block; an MPEG2 or MP3 decoder; an implementation of data encryption or decryption such as a Digital Encryption Standard (DES) block, or some other complex function[1].

0-7803-6677-8/01/$10.00©2001 IEEE.

Recently, the *platform-based design* approach to SoC designs has emerged [4,5,6]. This approach arose in consumer applications such as wireless handsets [7] and set-top boxes [8]. Elaborating on concepts presented in [9], we can define a platform as a co-ordinated family of hardware-software architectures, that satisfy a set of architectural constraints, imposed to allow the reuse of hardware and software components. We call this a "system platform". On a more pragmatic basis, platforms are collections of HW and SW IP blocks together with an on-chip communications architecture (on-chip buses, bridges, etc.) which usually include at least one processor, real-time operating system (RTOS), peripheral interface blocks, possible accelerating hardware blocks for specialised functions, middleware, and the option to customise the platform for specific applications through drawing HW and SW IP blocks from libraries. Recent work in highly programmable platforms, consisting of reconfigurable logic and fixed processor cores, has focused attention on embedded SW issues.

Platforms have moved beyond their wireless and multimedia focus to include, for example, automotive applications [10] and wired communications, including network processing.

One key question worth answering is: why platforms as a fundamental approach to SoC? Essentially, there are only a limited number of fundamentally "good" design solutions to the problems posed by a particular application area. A platform captures one or several related "good" architectures, which are optimal for an application domain, and allows their effective reuse in a low-risk manner with rapid time-to-market [11]. From the base platform, a number of derivative SoC designs can be created rapidly and with far lower effort than with a block-based approach. Another fundamental question is one of economics: who pays for what in creating a platform and its design infrastructure – the semiconductor provider, the IP house (especially a processor IP house) or the systems company? The future answer is quite unclear, although to date most investment has come from semiconductor and IP houses [11].

Notable industrial examples of platforms include Philips with its Nexperia Digital Video Processing platform design, Tality with Bluetooth designs, the BOPS voice-over-IP platform, the Metaflow Implosion SOC platform, Infineon M-Gold and TI OMAP wireless platforms, Parthus Mediastream, Palmchip CoreFrame, and reconfigurable platforms from Lucent (Agere), Improv Systems, Triscend, Altera Excalibur, Xilinx Vertex II, and Chameleon.

Design Methodologies, Processes and Flows

SoC design methodology incorporate a large number of related disciplines:

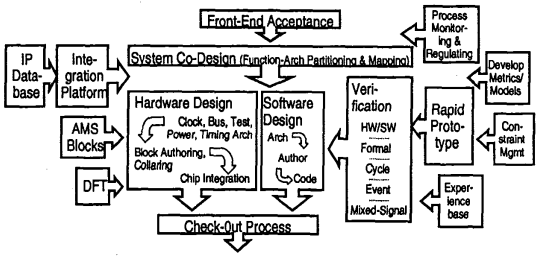

Figure 1: SoC Design Disciplines

In this tutorial paper we can only touch on them, highlighting areas of particular importance. These include:

- Design Process
- System Design
- Hardware Chip Design
- Functional Verification
- Analogue/Mixed-Signal
- Infrastructure

Design Process

This encompasses a set of basic processes to reduce the risk of SoC design through the systematic collection and reuse of design experience. The processes include:

- Mechanisms for logging the design process, comprising:
  - Metrics for measuring the design and design progress
  - Design sign-in points in the design flow
  - Efficient capturing of designer decisions and designer reasoning
  - Storage and retrieval of relevant design experience
- Certification to ensure completeness of design process
- Qualification to ensure the sufficiency of design decisions
- Process to use design experience for refining design flow

Such processes are important in order to increase confidence in taking on an SoC design project; to ensure effective reuse and minimal design time; to ensure that design experience is logged, and used for systematic improvement of SoC design processes and methods; and to allow quick assessment of the feasibility of a particular SoC design project. Further details on some of these processes can be found in [12].

System Design

System-level design is fundamental to effective SoC design, and is particularly effective when married with platform-based design concepts [13]. The essential concept is to allow exploration of design derivatives at the system level of abstraction, rather than the more traditional RTL and C level. Function-architecture co-design is a modern form of design space exploration that goes beyond hardware-software codesign, to permit a wider range of design tradeoffs. When platforms are modelled at the system level with appropriate support for design space exploration, the creation of a derivative design and its validation can be done rapidly, with a high probability that the derivative design implementation will be successful [14].

Hardware Chip Design

In the SoC era, for both block-based and platform-based methods, the emphasis is on chip integration, not the individual customised design of each block. In the platform-based approach, it is typical to design the "hardware kernel" or "foundation block" of the platform as a pre-laid-out, customised and characterised reusable element which forms part of every derivative design. The fixed hardware kernel generally incorporates the processors, processor and system buses, memory interfaces, possibly some fixed portion of memory, and bridges to peripheral bus. The platform-based design stage also usually fixes some part of the major SoC physical architectures: timing, test, power, etc. In the derivative design process for a platform-based design, the hardware kernel is assembled together with new, modified or pre-characterised blocks from IP libraries to form the final hardware realisation. Of course, the blocks to be assembled need to be designed in the context of the overall SoC design plan and constraints.

The major steps required in hardware design include:

- Chip planning
- Block design – using RTL synthesis and automated place and route
- Analogue/mixed-signal block design
- Memory block design
- Processor core design and assembly
- Bus design
- Library verification
- And finally, chip assembly

The most important step is chip floorplanning for both block-based and platform-based designs. The overall chip floorplan dictates the ease of block design and assembly, and imposes the major constraints on timing, power, test, etc. Developments in hierarchical chip design tools offering advanced floorplanning, estimation and assembly capabilities for block-based approaches are fundamental for SoC.

Functional Verification

It has been repeatedly stated in recent years that complex IC designs split into approximately 30-40% design effort and 60-70% effort on verification. SoC design with its additional system-level complexity only compounds this problem. However, a carefully chosen hierarchical verification strategy involving modern tools can go a long way to ameliorate this problem. The maze of current and new verification approaches, going beyond simple simulation, can be overwhelming for users to understand. A good guide to SoC functional verification is found in [15]. Here it will suffice to say that the emphasis in verification for SoC is a divide and conquer strategy that exploits the underlying hierarchical nature of the SoC design; and a transaction-based verification strategy that moves testbench stimulus, response and checking up from the Boolean signal level to more complex data types and system level transactions.

When verification strategies map onto the underlying SoC hierarchy, it is possible to verify designs block by block using a unit test concept, and then using selectively either the full block model, or a bus-functional model equivalent, within the overall SoC verification model. Testbenches similarly can be built on a unit, block basis, and reapplied within the overall SoC model. This is made much easier if the testbenches are architected to use the notion of high level transactions that are "natural" for the SoC. For example, processor and peripheral blocks communicate with on-chip buses via various bus read/write and acquire/release transactions, and it is easier to create testbenches that describe tests at this level and "protocol-convert" them to bit-wise signals, than to write them all at the bit-wise level. In addition, this greatly facilitates testbench reuse.

In the platform-based approach to SoC, reuse of testbenches and verification models is a given, because

the verification environment is first built for the platform, and then reused with little modification required for each derivative design. The only modifications needed are for any new IP blocks used, or for any blocks with modified functionality. Platform-based design allows the maximum amount of verification reuse when compared to a more custom, block-based SoC approach.

Analogue/Mixed-Signal

Although an increasing number of SoCs are mixed-signal, the bulk of them today are still heavily digital (we will discuss more advanced mixed-signal devices under "Advanced Design Concepts"). For the "D/a" designs, i.e. "big-digital/small-analogue" designs, containing primarily digital devices with analogue interfaces, the SoC design approach is primarily an integration approach. Analogue/Mixed-Signal (AMS) blocks must be designed for ease of integration onto a primarily digital device, and the chip assembly process needs to encompass methods for easy integration of these blocks.

The Virtual Socket Interface Alliance (VSIA) has done fundamental pioneering work in defining the standards for AMS block development and integration into SoC [16]. These standards define sets of deliverables for AMS blocks and classify them into mandatory and recommended subsets. AMS design issues act as an overlay on the hardware SoC design issues and tasks discussed earlier, with suitable modifications to handle the characteristics of the AMS blocks such as their susceptibility to noise and other parasitics. Today's best practices for AMS block authoring are based on a top-down, constraint driven approach [17]. Integration of AMS blocks designed with this top-down approach, and with VSI AMS standards in mind, can then become a low-risk and reliable standardised process. Clearly, this approach only works for "D/a" circuits dominated by the digital logic.

Infrastructure

Design information databases for IP blocks, platforms, and SoC designs continue to evolve. Reuse of design requires a well-architected and stable IP infrastructure in which design blocks (or platform aggregates of blocks) can be stored, searched for, found and reliably retrieved. But the database infrastructure is only a foundation or substructure. Of equal importance are a set of IP management processes and procedures that allow effective use of the underlying database capabilities. Here, effective characterisation and rapid search for IP blocks and platforms is a key requirement.

Embedded Software Development for SoC

The importance of embedded software for SoC design grows and grows [18]. Embedded SW clearly is different than applications software on a personal computer (PC). PC software is not real-time and offers no guarantee of responsiveness. Reliability and ease of use have not been issues, especially considering the history of Windows and applications running on it. PC software has bloated considerably over time, relying on the underlying HW platforms (the PC platform moving from x86's to Pentium I, II, III and beyond) to offer the vastly increased disk space, memory size and processing power in order to keep pace with software feature bloat. Since overall software size and runtimes are not particularly critical, applications developers have been able to use the latest tools and abstractions (Object-Oriented Programming, Visual C++, for example) in order to achieve productivity. Finally, since the underlying platform is relatively fixed or moves in slow increments with reasonable amounts of backwards compatibility, porting of PC software is not a huge issue, although still a source of pain.

Contrast this with embedded software in realtime systems. Here, real-time characteristics are critical. Constraints range in importance from weak to very strong, and deadlines are often very hard, especially for safety-critical systems such as automotive that demand guaranteed responsiveness. Consumer demands for reliability and ease of use are much more important to the success of an embedded product. Cost considerations imply very tight constraints on software size, latency and responsiveness, with extreme attention paid to memory size and footprint, processor class, and assembly coding is still very much used especially for DSPs to guarantee performance. Finally, porting of highly-tuned SW from one platform to another is very much an issue.

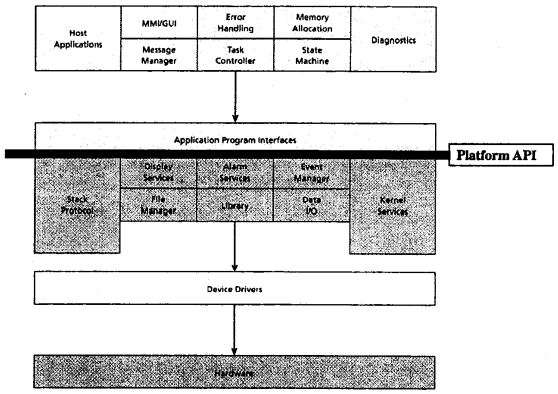

Figure 2: Architecture of Embedded SW

Figure 2 illustrates one possible architecture or layering

for embedded software for a platform-based design approach. The most important concept shown here is not the specific layering, but the concept of a 'Platform API' which hides the details of the underlying HW architecture, and the low-level SW, from the applications designer, without limiting their ability to optimise the resulting code. The whole area of effective mechanisms for embedded SW development in a platform-based SoC context is under active study [19,20].

Advanced Design Concepts for SoC

Recent thinking in the area of SoC has developed concepts in several areas. The first lies in what we might call "The Superchip": moving from the digital-dominated mixed-signal SoC to one where analogue, digital, and indeed other processing domains (such as RF, MEMS, Optical, etc.) are all considered equal partners for design and integration of complex systems. The idea of a Superchip brings with it a host of new trade-off possibilities. Key challenges of the superchip are the need to integrate and communicate between what have been very disparate design disciplines (digital to analogue, system to IC, IC to package/board design). Of course, the requirements of each design domain continue to evolve rapidly. The move to the true "A/D" "superchip" will need to be made incrementally with refined tools and methodologies, as they become available.

To illustrate, let's consider the kind of co-design across disciplines required with an example. In the area of functional verification, digital designers have moved to a top-down, language-centric, synthesis based design methodology with standardised Hardware Description Languages and compatible simulators. Speed and capacity of digital simulation are key issues. On the other hand, analogue designers have only gradually begun a migration from a bottom-up, transistor-centric design methodology, in which accuracy and capacity of Spice/Spectre/RF-type simulators are key drivers. Digital designers are experimenting with C/C++ language based design, but the take-up of analogue behavioural modelling is much slower. Bringing these two types of design teams together and offering a compatible co-design and co-verification environment requires the support of both top-down and bottom-up methodologies, new variants of the standard languages such as Verilog-AMS and VHDL-AMS, capacity, speed and various accuracy tradeoffs well understood, and equal attention paid both to cell library models (and newer IP block models) and semiconductor device models.

Another advanced design area that is rising in importance is to treat on-chip communications between IP blocks or islands of functional processing not as simple point-to-point connections, or the use of standard board-style buses, but as the design of an on-chip communications network closer to that in the telecom/datacom networks of today [21].

A final area to discuss is the rise of new SoC implementation fabrics, especially the highly programmable platforms based on a combination of fixed processor cores and reconfigurable logic. As ASICs add reconfigurable logic regions for maximum flexibility; as FPGAs become "platform FPGAs" with fixed processor and memory cores; and as new reconfigurable logic architectures emerge, from companies such as Adaptive Silicon, Chameleon, and others, the boundary between the ASIC and FPGA design styles begins to blur. Ultimately, a hybrid of the two seems destined to become the optimal implementation fabric for a host of SoC applications. Perhaps a more interesting question is where the dividing line will be drawn between the hybrid, highly programmable platforms, and the more cost-optimised, less flexible, custom design approaches to SoC. Clearly some applications may never lend themselves to this approach but high economic rewards will accrue to those who choose the right path in each domain.

Conclusions

In this tutorial paper we have covered many aspects of modern SoC design, both current issues and advanced design aspects of the future. The challenges are large but are being met by many design teams around the world. A key decision to be taken is the basic approach to SoC, and it is here that we recommend strong consideration of platform-based approaches. If they are right for the application area, they offer the highest degree of reuse and designer productivity. For other areas, the hierarchical block-based design approach is best. Key areas, which must be addressed by SoC methods, include embedded SW, mixed-signal design, communications design, and the choice of implementation fabrics.

Acknowledgements

This work spans many areas of expertise and is a high level overview of over two years of development work within Cadence Design Systems on an SoC Platform Based Design Methodology. Key technical architects included: Larry Cooke, Antoine Goujon, Merrill Hunt, Wuudian Ke, Christopher Lennard, Grant Martin, Peter Paterson, Khoan Truong, and Kumar Venkatramani. We would also like to acknowledge the entire development team as well as the other authors of *Surviving the SoC*

Revolution: A Guide to Platform Based Design [1].

References

[1] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd. *Surviving the SOC Revolution: A Guide to Platform-Based Design*, (Kluwer Academic Publishers, Boston, 1999), p. 4

[2] G. Martin, H. Chang, L. Cooke, and A. de Oliveira, "Introduction to Platform-Based Design of Systems on Chip," *IP/SoC 2001 Tutorial*, (Santa Clara, California, March, 2001)

[3] G. Martin, B. Jackson, L. Pandula and M. Nuotio, "Key components of a wireless terminal platform architecture", *Proc. Of Mobile Handsets 99*, (London, May, 1999), Day 2, Stream A, p. 9

[4] A. McNelly, L. Todd and G. Martin, "Application platform speeds up IP integration", *Electronic Engineering Times*, p. 60,70 (June 15, 1998)

[5] G. Martin, A. McNelly and L. Todd, "The Integration Platform Approach to System-On-Chip Design", *Proc. of IP 98 Europe*, (Frankfurt, Germany, 1998), p. 101

[6] R. Quinnell, "Platforms for Integration", *Silicon Strategies*, (January, 1999), p. 10

[7] G. Martin, "Wireless takes an integrated platform approach to System-On-A-Chip designs", SystemChips Technology and Applications, Supplement to *Wireless Systems Design*, 8-11 (August, 1998)

[8] T. Claasen, "High Speed: Not the Only Way to Exploit the Intrinsic Computational Power of Silicon", *ISSCC 1999 Digest of Technical Papers*, (February, 1999), p. 22

[9] A. Sangiovanni-Vincentelli, and A. Ferrari. "System Design - Traditional Concepts and New Paradigms", *Proc. Of ICCD 99*, (Austin, October 1999), p. 2

[10] A. Ferrari, S. Garue, M. Peri, S. Pezzini, L. Valsecchi, F. Andretta, and W. Nesci, "Design and Implementation of a Dual Processor Platform for Power-train Systems", *Proc. Of Convergence 2000*, (Detroit, October, 2000)

[11] W. Wolf and J. Henkel, "Platform-Based Design", *Tutorial at DATE 2001*, (Munich, March, 2001)

[12] C. Lennard, C. and E. Granata, "The Meta-Methods: Managing Design Risk during IP Selection and Integration", *Proc.s of the Intellectual Property System on Chip Conference*, (Edinburgh, November 1999), p. 285

[13] G. Martin, "Productivity in VC Reuse: Linking SOC Platforms to Abstract Systems Design Methodology", *Proc. Of Forum on Design Languages: Virtual Components Design and Reuse*, (Lyon, August-Sept. 1999), p. 313. Also found in Chapter 3 of: R. Seepold and N. M. Madrid, editors, *Virtual Component Design and Reuse*, (Kluwer Academic Publishers, Dordrecht, 2001)

[14] F. Schirrmeister and G. Martin, "Platform-Based Design Helps meld EDA with Convergence Demands", *Wireless Systems Design*, p. 21, (May 2000)

[15] P. Rashinkar, P. Paterson, and L. Singh, *System-on-a-Chip Verification: Methodology and Techniques*, (Kluwer Academic Publishers, Boston, 2000)

[16] Virtual Socket Interface Alliance web site: URL http://www.vsi.org/

[17] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, E. Malavasi, A. Sangiovanni-Vincentelli and I. Vassiliou, *A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits*, (Kluwer Academic Press, Boston, 1997)

[18] G. Martin, and C. Lennard, "Improving Embedded SW Design and Integration for SOCs", *Proc. Of CICC*, (Orlando, May 2000), p. 101

[19] K. Keutzer, S. Malik, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System-Level Design: Orthogonalization of Concerns and Platform-Based Design", *IEEE Trans. on CAD of ICs and Systems*, **19, 12**, 1523, (December 2000)

[20] G. Martin, L. Lavagno, and J. Louis-Guerin, "Embedded UML: a merger of Real-time UML and co-design", *Proc. Of CODES 2001*, (Copenhagen, April 2001), p. 23

[21] Session 41, "On-Chip Communication Architectures", *Proc. Of DAC* 2001, (Las Vegas, June, 2001), p. 667+