

Section 1.1

System on Chip

The Challenge and Opportunities

Wael Badawy

Dept. of Electrical and Computer Engineering

University of Calgary

Calgary, Canada T2N 1N4

Abstract: The increase in the number of logic gates (10 million to 20 million gates) that can be implemented on a single integrated chip allows for design of circuits with great functional complexity. A whole system previously integrated into a board is now being integrated into a single chip leading to System on Chip (SoC) design. A paradigm shift from the current cell library-driven application specific integrated circuit (ASIC) design to a new System on Chip (SoC) technology is now occurring. This paper outlines the challenges in the development of SoC design for real time systems.

Keyword: System on Chip

1. INTRODUCTION

With the increase in the number of logic gates that can be implemented on a single chip, various functionalities, like memories, logic, analog, CPU, DSP, MPEG decoder/encoders, etc. can be integrated into a single silicon chip, realizing an entire system on chip (SoC). According to the virtual socket interface (VSI) alliance, SoC is defined as a highly integrated device. It is also known as a system on silicon, a system-on-chip, system-LSI, system-ASIC, and as a system-level integration (SLI) device [1]. An SLI device is defined as “greater than 100 thousand gates with at least one programmable core and on-chip memory”[2]. Another definition can be found in [2]: “A complex IC that integrates the major functional elements of a complete end-product into a single chip or chipset” In this paper the SoC is defined

as a complex IC that consists of one or more programmable cores, connected using an on-chip bus with signal processor cores (digital or analog), and that may incorporate on-chip memory. The proposed definition describes the difference between a SoC microprocessor and as well it distinguishes SoCs and complex application specific integrated circuits (ASICs).

With the rapid achievement of mass production volumes in consumer products, SoC development faces a continuous decrease of the time to market window. It is expected that in the near future new consumer products will take less than one year to achieve a market penetration of ten million units. All the components that today are part of a system will have to be integrated together. Therefore one of the key factors for the success of ultra-deep sub-micron technologies will be the capability of integrating into a single piece of silicon different components, such as logic functions, analog functions, and memories of different types (static memories with either dynamic or non-volatile). Design methodologies will evolve and Concurrent Engineering, Intellectual Properties (IP) reuse, HW/SW co-design and co-verification, fast prototyping and new CAD tools will be common practice in the coming years. SoC as an emerging technology has its own challenges, issues and trends. This chapter outlines the issues, challenges and trends for the new SoC era in the key areas out lined in Figure 1.

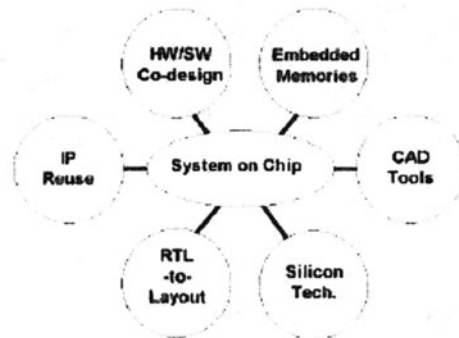


Figure 1. The main challenges and trends in SoC development

2. TECHNOLOGY CHALLENGES

SoC requires designing and integrating digital modules with analog blocks, special memories and RF designs for some applications. Due to the design complexity of the SoC, its architecture is based on the reuse of the Intellectual Property (IP) core to meet time-to-market requirements. Virtual Component (VC) or IP modules are often originated from different vendors with different formats.

Virtual Socket Interface Alliance (VSIA) [3], Virtual Component Exchange (VCX) [4] and Reusable Application Specific Intellectual Property Developers (RAPID) [5] are used to define the IP cores that allow exchange between vendors.

The Reuse Methodology Manual (RMM) and Open MORE (Measure Of Reuse Excellence) [6] developed by Mentor Graphics Corp. and Synopsys Inc. [7] provide guidelines for design, test-benches, model creation and physical integration [8]. These guidelines provide the underlying methodology to establish a best practice IP reuse process for SoC designs.

Modular processing for implementing SoC designs starts with a high-speed digital CMOS process followed by processing masks to implement technologies such as high density SRAM, DRAM, flash memory, and BiCMOS for RF applications. While digital design among different vendors is quit similar, adding analog capabilities to a baseline digital process has a huge overhead since the process is different among vendors.

Feature size reduction is the driving force for device scaling, which has led to increased integration. However this scaling raises short channel issues [9] in transistors. The process of integrating analog components like ADCs, DACs, and PLLs along with digital ones is not only very complex, but it will also be extremely difficult to obtain a good yield ratio. An alternative solution is to use a multi-die approach where different functions are confined to different dies, diffused in different processes and later combined in one package. Issues in the fabrication of SoCs are derived from the mixing of each individual technology developed separately for each application. For correct functionality, MOS scaling requires gate oxide thinning and reduction in supply voltage. As a result of the increasing gap between transistor and I/O voltage, and also because many different voltages have to be considered in the same design, innovations in process technology have become necessary [10]. The multiple gate oxide solutions, which was first used for non-volatile memories are now used in the CMOS process. Introduction of multi-gate oxide (dual and triple) solutions are the most recent process modifications for SoC architectures [11].

Among the technological challenges is the development of multiple threshold voltage transistors that require leakage management strategies for high performance SoCs. Moreover, a thick gate oxide must be developed that can sustain the high voltage required by different embedded memory components. Embedded memories consume the most power and the largest area of SoC

3. DESIGN METHODOLOGIES

Embedded telecommunications and multimedia define new design challenges such as heterogeneity, dynamic behavior and dynamic memory management, architectural and timing problems due to IP reuse, changing requirements and

compatibility to a number of standards. Integrating both HW and SW components and the ability to perform complete system simulation rapidly and accurately is one of the main issues for the SoC development. SoC changes the design space from ASIC cores to platforms for running embedded software. Most embedded real time systems have some kind of Real Time Operating System (RTOS) or programmable kernel. An RTOS can be described as multi-threaded and preemptable, supporting thread priority and synchronization. Multithreaded processor cores are required to handle interrupts and maintain the required throughput [12]. The interrupt latency and context switching time depend on the application and should be accurately modeled

High level modelling languages

Due to the complexity of the SoC, new design process with higher level of abstraction to capture both hardware and software are required to model the hardware/software interaction. New languages with greater reliance on tools are required to express the interaction of hardware and software components.

New classes are added to C++ that allow concurrency and reactivity to describe hardware/software interactions [13], [14]. This library extension approach allows for the development of a high-level system model, which captures functionality, structure and timing. It allows refinement to proceed from a functional model to a state machine model and finally to a synthesizable RTL model. To describe hardware using programming languages requires Concurrency, Instantiation, Reusability, Reactivity, Hardware data types and operators, and HW/SW co-verification.

- **Concurrency** - Hardware is essentially concurrent i.e. all cores execute in parallel with respect to each other. Concurrency is implemented using parallel threads or an event queue.

- **Instantiation** – Architecture is modeled into simpler components with less complexity, which can be effectively managed. Instantiation creates a hierarchy of components (objects) where each component runs concurrently and emulates the parallelism in a circuit.

- **Reusability** - Creation of smaller, less complex kernel components allows for reusability.

- **Reactivity** - The ability of a functional block to get executed based on the occurrence of a stimulus. Reactivity allows the modeling of combinational elements that react to the clock edges, which allow modeling of sequential elements.

- Hardware data types support flexibility to reflect the size of data (similar to 'length in VHDL) and operations. Operators such as concatenation, bit selection and non-blocking assignments are required to describe hardware effectively.

- Using high-level languages allow HW/SW co-verification. Reusability is done by the constructor of a class that defines the interface of the class with other parts of

the design. The definition of an IP can be encapsulated within a constructor. This means the users of the IP are provided with the constructor. This enables the users to create an instance of the IP within their design along with the interfaces.

SystemC [15] and SpecC are popular languages for system level design. SystemC is a modeling platform consisting of C++ class libraries for system level design. The main goal of using SystemC is to promote and accelerate the exchange of IP models and executable specifications using C (C++) based platform. Through an open source language and reference implementation, designers can create, validate and share models with other vendors. SpecC, originally developed by UC Irvine, is a language-specification description language based on C designed to support integration of product development, from determining specifications to implementing the design. SpecC methodology is an IP centric codesign methodology for specifying, modeling and designing embedded and SoC systems at system level. Moreover, System Level Design Languages (SLDLs) including Dataflow style languages (static and dynamic) which are used for signal processing applications can be used for embedded systems and control dominated languages such as StateCharts, CoCentric System Studio and Esterel [16].

Platform Based Design

Platform Based Design (PBD) is used for rapid SoC development. A stable core-based architecture can be rapidly extended and customized for a range of applications. SoCs have to be designed with flexibility in mind to meet changing standards and market trends. Changes required for the next generation SoCs must be made quickly and efficiently to meet several market requirements. PBD takes the best out of HW/SW co-design. Instead of designing SoC architectures from scratch, architectural platforms are developed for particular applications. Integration of IP cores is used to customize system components with both hardware and software to suit different applications.

CoWareN2C [17] is a platform with many derivatives of a core that can be customized by adding more HW and/or SW. PBD enables designers to rapidly turn around platforms for many derivatives for a particular product line. In PBD, design reuse is planned for a specific family of products and is based on pre-verified blocks with standard interfaces. CoWare methodology separates function from communications, making the replacement of one piece of IP with another easier. It also allows the evaluation of the potential of one piece of IP versus another. CoWareN2C can also be used to create virtual platforms for integrating HW and SW using high-level languages like SystemC.

The main issue with PBD is concerned with providing a platform that satisfies the needs of a broad base of designs while providing capability for customization and product differentiation.

Cost evaluation will be used to analyze performance benefits of different modifications. Creating new architectures may require changes in software.

Dynamic binary translation will be used in order to avoid expensive and time-consuming software re-design. Binary translation techniques aim at translating object code (binary code) from one architecture to another [18]. Dynamic binary translation uses a translator to execute the program, i.e. the input program is dynamically analyzed and translated to target code. Dynamic binary translation overcomes some of the disadvantages of static translation. Some form of hardware wrapper will probably be required for dynamic translation in SoCs.

Memory management

One of the main performance challenges in computer systems today is the speed mismatch between microprocessors and memory. Processors use a large fraction of the transistors and area of the chips for large SRAM caches. In SoC designs, memory is the largest part of the system and it may be about 70-90% of the chip. Memory management is one of the critical design processes due to the limited bandwidth. SoC solutions will narrow this processor-memory performance gap due to the absence of off-chip bus transfers. In traditional systems, the operating system is responsible for all background memory and storage related tasks. On the other hand, embedded systems cannot use software run-time memory management since it is expensive in terms of area, performance and power. Moreover, embedded systems use dedicated distributed memory systems. Therefore, dynamic memory management must be synthesized in the embedded system implementation itself. Matisse [19] is a design environment that optimizes memory management. It allows the exploration of different architectures to select optimal memory management design.

Chinook [20] and Polis [21] and data flow dominated methodologies such as COSSAP (Commercial DSP synthesis tool from Synopsys) and Ptolemy [22] are frameworks for HW/SW co-synthesis and include control dominated methodologies like Telecom applications and hardware that involve lots of protocol processing. These applications are modeled as a set of concurrent processes that access data. Concurrent object-oriented models play a key role in large scale HW/SW system design for these applications.

Intelligent RAM (IRAM) or Computational RAM (CRAM) [23] offers several advantages such as less latency, increased bandwidth, reduced power/energy consumption, and reduced space and weight for embedded portable systems. In SoC design, issues such as scheduling, resource allocation, memory hierarchy and organization, manipulation and modification of stored data structures need to be addressed in terms of explicit memory I/O operations.

Power consumption

With the advances in the mobile communication and personal communication services (PCSs), there is increasing demand to run video applications on mobile

systems. High-speed data and computation-intensive processing will lead to much higher power consumption than traditional portable applications. Due to the limited power-supply capability of current battery technology, current research efforts face two conflicting design challenges. One is to explore high-performance design and implementation techniques that can meet the stringent speed constraints for real-time systems. The second is to consider low-power design approaches to prolong the operating time of portable devices.

Power consumption has been introduced as a major design parameter due to the rapidly advancing technology of mobile communications, which require portable power supplies. There are three main sources of power dissipation in a CMOS circuit: (1) switching power, which is consumed by the circuit capacitance during transistor switching; (2) short-circuit power, which is due to current flowing from power supply to ground during transistor switching; and (3) static power, which is due to leakage and static currents. The switching and short circuit power are known as dynamic power, which constitutes about 99% of the total power dissipation in CMOS VLSI circuits. Dynamic power, $P_{dynamic}$, is given by the following equation:

$$P_{dynamic} = f_{clk} \cdot V_{dd} \cdot ((C_i \cdot V_{swing\ i} \cdot p_i) + I_{sc} \cdot p_i)$$

where f_{clk} is the system clock frequency, V_{dd} is the power supply voltage, C_i is the capacitance at node i , $V_{swing\ i}$ is the voltage swing at node i , p_i is the switching probability at node i , and I_{sc} is the short circuit current at node i .

As system speed increases, the increase in power consumption becomes a major problem. The increase in the power dissipation will generate extra heat and reduce the reliability of a circuit. The development of techniques allowing for the development of low-power circuits is crucial to the practicality of future products. Due to its quadratic dependence, lowering V_{dd} contributes a significant power saving. However, the delay in processing time will increase as V_{dd} decreases. To meet the throughput requirements, the delay must be compensated for at the system and algorithmic/architecture design level. Simply, a new version of the design that allows more latency is required.

Low-power system-level approaches involve two main techniques. One is partitioning the system into modules and shutting down modules that are not working such as multi-chip modules (MCMs) [24]. The other approach is to use a monitor scheduler to distribute the workload over several functional units. The scheduler can be a hardware or software monitor.

Besides the system-level low-power approaches, the algorithmic/architectural approach is the most promising one [25]. Reformulating the algorithms and mapping them to efficient low-power VLSI architectures to compensate for the speed penalty caused by low supply voltage can achieve the algorithmic/architectural low-power design. The power savings of the

algorithmic/architectural level approach are in the range of 70-90% [26]. The current goal is to reduce the total power dissipation of the electronic systems to two orders of magnitude less than what it would have been with the conventional technology [27]. Approximations, parallel and pipelined design approaches are used to compensate for the delay penalty [25], [28].

Low-power and high-performance architectures can be achieved using approximation approaches which trade high performance with the cost of the circuits, pipelining which trades the overhead of extra buffers with latency, and parallel architectures which trade area with latency.

In the approximation approaches, the constraints are relaxed in order to reduce the cost and yield acceptable performance. Approximation approaches generally lead to much simplified architectures, which then add to the savings in total implementation cost. For example, three-step motion estimation [29] is usually employed to reduce the computational complexity while maintaining a performance comparable with full-search motion estimation schemes [30].

Pipelining is a commonly used technique to achieve high-speed data processing. The main idea behind pipelining a design is shortening the delay through the critical path by inserting latches between consecutive pipeline stages. As a result, the speed of the system is faster than that of the original system at the penalty of increasing the latency by the number of pipelining stages. On the other hand, pipelining can be used to compensate for the delay incurred in the low-power design when the supply voltage drops significantly. Some derivations of the pipelining techniques are asynchronous circuit pipelining [31], two-level (word/bit-level) pipelining [32], and bit-level pipelining approaches [33].

Parallel data processing decomposes the desired function into independent and parallel small tasks. These tasks are executed concurrently, and individual results are combined to obtain the desired computation result. It is a "divide-and-conquer" strategy in solving problems. The Fast Fourier Transform (FFT) [34] is a famous example where the algorithm first decomposes an N-point FFT into two N/2-point FFTs, and expands the computation recursively to form a parallel computing network.

Power minimization for real time SoCs is addressed at the level of several synthesis tasks including scheduling, instruction and data cache size determination (application specific) and processor core selection. Voltage is treated as a variable. Non-preemptive scheduling of a set of independent tasks with arbitrary arrival times and deadlines on a variable voltage processor is a NP-complete task [33].

Power consumption is a new set of design requirements due to the limitation of the current battery technology. New methodologies for low-power core-based real-time SoCs are based on dynamic variable voltage. Effective scheduling techniques treat voltage as a variable whose value is to be determined in addition to conventional task scheduling and allocation [35]. Conventional methods for reducing power consumption in a core in CMOS technology involve lowering the

supply voltage level, which exploits the quadratic dependence of power on voltage. Reducing the supply voltage however increases the circuit delay and decreases clock speed. This technique is therefore ineffective when tight deadlines are present in systems.

Networked SoCs such as pocket computers, wireless sensors, pagers and cell phones rely on efficient use of power, reliability and thermal management. These devices have a much wider dynamic range of power demand. Power consumption is of two types - instantaneous and average power. Average power is affected by communication protocols and power management strategies. Instantaneous power is affected by transceiver architecture and RF circuit design. The signaling protocols, choice of modulation, transceiver architecture and RF (IF) circuit design affect power. Power management is done at different modes of operation such as transmit, receive, idle, sleep and off modes. Effective power management requires power aware protocol design.

Reconfiguration in SoCs

Reconfigurable computing achieves significant speed on a general class of applications [36]. For SoCs, different configuration bit streams will be stored on chip. It will introduce overhead in the amount of storage, which should be minimal. Moreover, dynamic reconfiguration at run time introduces the overhead of the reconfiguration time. Multiple reconfigurable bit streams are stored for different contexts, which consumes memory space. In SoC implementations, the entire application needs to be stored in on-chip memory. A solution that minimizes this configuration overhead using distributed caches is presented in [37]. The flexibility requirement of SoC requires programmable devices with low power that support multiple standards of communication, audio and video. Traditional DSPs and processors offer high flexibility but are power inefficient, as opposed to ASICs, which are not flexible but are efficient in terms of speed and power. An architecture framework that is a tradeoff between ASICs and programmable processors is the Pleiades, which is a low-power reconfigurable DSP architecture composed of heterogeneous components [38].

4. ANALOG COMPONENTS IN SOC

Analog and RF components along with digital ones will be the main components of SoC. The 3G mobile telephony is one of the main technology drivers for mixed signal processing and EDA tools. High performance demands and complexity at reduced cost are driving the integration of RF and analog components into digital designs. RF, digital and analog circuits have different design methodologies and therefore integrating these on mixed signal ICs is quite difficult. Some problems in integrating analog and mixed signal IP onto dies with logic functions include noise

issues, dissimilar power requirements and actual implementation between two separate engineering organizations. IP core and packaging address only digital logic while many of the new ICs have analog and mixed signal interfaces. The main challenge is that analog space cannot be fully covered by any finite library of blocks. Analog components are often customized cells. Simulation-based synthesis is a new method that is proposed to synthesize a range of custom analog cells large enough to be compatible with industrial designs although the cost of a full circuit-level simulation of a design candidate is not practical [39].

The analog components exchange data with digital ones using A/D converters. One of the challenges of low power is that transducers usually generate signals that are in the microvolt to milli-volt range that is well below the resolution of a 10-bit A/D converter. Therefore, direct digitization is not possible. In addition, many digital applications like hard-disk servos exhibit signal characteristics those require extensive analog processing prior to the digital operation. The latency and delays to convert the analog signal into a digital one introduce a delay overhead.

Moreover, the proximity of analog and digital devices on the substrate create increasingly difficult design challenges. In a high accuracy and precision system, the analog circuitry is placed on a separate chip. To protect analog blocks from digitally induced noise, a SoC process must isolate analog and digital blocks. This introduces extra space between the blocks that increases the area; placing an oxide barrier between them; or surrounding them with heavily doped isolation rings. Analog cores are more susceptible than digital ones to thermal problems. Isolation of analog blocks from heat producing digital blocks such as clock generators or output buffers is necessary.

Using copper instead of aluminum metallization results in lower total chip power dissipation and smaller on chip temperature rise reducing analog component drift [37]. New tools are being developed to design and verify mixed signal circuits. ADVanceMS [38] supports top-down and bottom-up design methodologies for mixed signal SoC. It utilizes IEEE standard 1076. ADVanceMS uses capabilities of ModelSim and Eldo (a Mentor Graphics product for analog transistor simulation) for digital and analog circuit simulation respectively. Moreover, ADVanceRFIC extends features of Eldo to perform frequency domain RF analysis for mixed signal designs [40].

5. ON-CHIP BUSES

Traditional design methodologies when applied to SoCs lead to a very long design verification time due to the impact of "glue logic." Glue logic is the custom designed hardware linking all functional IP cores and setting up communication between functional cores and off-chip peripherals. The verification time becomes longer since the glue logic function has to be verified with all other functional units

connected. The interface functions and internal functions related to interfaces in IPs must also be verified [41]. To reduce the verification time of glue logic, SoC bus cores have been proposed by VSIA, IBM and other vendors. New bus architectures based on circuit and packet switching (different from classical buses) needs to be investigated. A solution to reduce the verification time is to remove all glue logic by using a bus controller for all components in the SoC.

Current hardware solutions for protocol processing are of two types -ASIC and programmable CPU. An ASIC-based solution is not flexible, and the performance of the programmable CPU is low. Network applications have to be processed in real time and the protocols have to be adaptive. Current hardware for such applications is not very good. Protocol processors that process framing-deframing on the fly is required. The IBM Coreconnects [42] eases integration and reuse of processor, system and peripheral cores within standard product and custom SoC designs. Other popular bus architectures are the C bus and Lexra Bus Controller (LBC) [43]. Another architecture is the Integration Architecture - SonicsIA micronetwork [44]. A micro-network is an on-chip network that decouples, unifies, and manages all communications between heterogeneous cores such as the microprocessor, digital signal processor (DSP), and/or I/O functions. There is no single bus architecture that serves as a universal solution for a SoC design targeted at communication applications.

The IEEE 1394 standard originally developed by Apple is a very fast external bus standard that supports data transfer rates of up to 400Mbps. In addition to its high speed, 1394 also supports isochronously data-delivering data at a guaranteed rate. This makes it ideal for devices that need to transfer high levels of data in real-time, such as video devices. Like USB (Universal Serial Bus), IEEE 1394 supports both Plug-and-Play and hot plugging. The main difference between 1394 and USB is that 1394 supports faster data transfer rates and is more expensive. For this reason, it is expected to be used mostly for devices that require large throughputs, such as video cameras, whereas USB will be used to connect most other peripheral devices.

6. SOC VERIFICATION

Verification of today's SoCs occur at low levels of abstraction, typically RTL. As the complexity of SoCs grows, it is important to move verification to higher levels of abstraction. Hardware/software co-simulation is a step in this direction but is not sufficient due to deficient processor models and slow hardware simulation speeds. It is also difficult to model the real-world environment around the SoC. New verification languages such as Vera from Synopsys, and RAVE from Chronology [45] have enabled easier and precise functional verification along with testbench generation.

Support for system-level verification already exists for the software part of SoC, where a common technique is to use software monitors to instrument code. Instrumentation of software utilizes target resources such as memory and execution time. When the instrumentation code is removed the system may change behavior due to timing differences. Hardware based monitoring makes a SoC observable at different abstraction levels in both hardware and software. A small hardware component is integrated into the SoC [43]. It works like a probe either listening to logic or system level events in a passive manner or is activated by software that writes to a specific register. When using a hardware based RTOS kernel for management of SoC software, the process-level events can be monitored without software overhead and with no intrusion on the systems' timing behavior. Since it is nonintrusive, it gives the SoC developer the free choice of either keeping or removing the monitor component in the final product. A hardware RTOS kernel implements traditional (software) RTOS functions (scheduling, task management, interprocess communication, synchronization etc.) in hardware. The RTU (Real Time Unit) is such a component that has proven to be successful for increasing RTOS performance and operational predictability [46].

Traditional test methods are not suitable for core-based designs since the core user is responsible for manufacturing and testing the core. The IEEE P1500 standard [47] addresses testing embedded core based systems while the IEEE ISO Nexus 5001 [48] forum addresses real time debugging. The IEEE 1149.4 [49] mixed signal test bus is being developed to support mixed signal built-in structures for improving controllability and observability so that test development time is reduced.

BIST is an appropriate method for testing complex systems. Deterministic BIST methods are particularly important as they do not require any modifications of the core under test thus protecting intellectual property. The main difficulties in testing SoC designs are due to reduced accessibility of the cores, increased complexity of the chip and heterogeneity of the modules. Testing SoC designs involves individual core testing and system testing. Test-per-scan and Test-per-clock are the most popular BIST schemes [50]. The Test-per-scan method uses a complete or partial scan path that is serially filled by the ATPG (Automatic Test Pattern Generator). The scan chain contents are applied to the Unit Under Test (UUT) and the response is loaded back into the scan chain in parallel. Then a new bit stream is shifted in and the scan output is compressed. In the Test-per-clock method, special registers are used which work in four modes. In the system mode, the registers operate as ordinary flip-flops.

In the pattern-generation mode, they perform autonomous state transitions, and the states are the test patterns. In the response evaluation mode, the responses of the UUT are compressed and in the Shift mode the registers work as a scan path. In the pattern generation mode, the registers and logic are configured as a linear feedback shift register. This scheme leads to shorter test times as a new pattern is generated in each clock cycle at least for a part of the circuit [50].

REFERENCES

- [1] VSI Alliance Glossary, VSI Alliance, March 98, www.vsi.org
- [2] H. Chang et al., *Surviving the SOC Revolution*, Readings, Kluwer Academic Publisher, 1999.
- [3] VSIA Alliance Architecture Document, Ver 1.0, www.vsi.org
- [4] Virtual Component Exchange, www.vcx.org
- [5] RAPID, www.rapid.org
- [6] www.mentorg.com/openmore
- [7] www.synopsys.com
- [8] S. Dey, C. N. Taylor et al., "Using a Soft Core in a SoC Design: Experiences with picoJava", *IEEE Design and Test*, July-September 2000, pp. 60-71.
- [9] J. P. Bruines, "Process Outlook for Analog and RF applications", *Microelectronic Engineering*, 54 (2000), Elsevier, pp. 35-48.
- [10] M. Haond, "Issues for CMOS Integration in Systems on a Chip", *Journal of Microelectronic Engineering*, 54 (2000), Elsevier, pp. 23-34.
- [11] P. Senn, "Telecommunication Applications: New Challenges for Microelectronics", *Journal of Microelectronic Engineering*, 54 (2000), Elsevier, pp. 5-14.
- [12] C. Zilles, J. Elmer, G. Sohi, "The use of Multithreading for Exception Handling", *MICRO-32*, Nov 1999, pp. 219-229.
- [13] D. Verkest, J. Cockx et al., "On the use of C++ for System on Chip Design", *Proc of IEEE Workshop on VLSI*, April 1999.
- [14] R. Roth, D. Ramanathan, "A High Level Hardware Design Methodology Using C++", *Fourth High Level Design Validation and Test Workshop*, 1999, pp. 73-80.
- [15] www.systemc.org
- [16] L. Blanc, S. Dissoubay, "Esterel Methodology for Complex System Design", *Microelectronic Engineering*, 54 (2000), Elsevier, pp. 163-170.
- [17] CoWareN2C for Platform-based Design, www.coware.com
- [18] K. Ebcioglu, E. R. Altman, "DAISY: Dynamic Compilation for 100% Architectural Compatibility", *IBM Research Reprt*, www.research.ibm.com
- [19] D. Verkest, J. da Silva et al., "Matisse: A System on Chip Design Methodology Emphasizing Dynamic Memory Management", *Journal of VLSI Signal Processing*, Vol 21, No. 3, July 1999, pp. 277-291.
- [20] P. Chou, R. Ortega et al., "The Chinook Hardware/Software Co-Synthesis System", *International Symposium on System Synthesis*, September 1995, pp. 22-27.
- [21] F. Balarin et al., "POLIS: A Design Environment for Control-dominated Embedded Systems", *Technical Report*, UC Berkeley, 1996.
- [22] J. Buck et al., "Ptolemy: A Framework for Simulating and Prototyping Heterogenous Systems", *International Journal of Computer Simulation*, special issue on simulation software development, Vol 4, 1994.
- [23] IRAM Project, <http://iram.cs.berkeley.edu>
- [24] S. A. Khan and V. K. Madiseti, "System Partitioning of MCM for Low Power," *IEEE Design Test Comput. Mug.*, vol. 12, pp.41-52, Spring 1995.
- [25] A. P. Chandrakasan and R. W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," *Proc. IEEE*, vol. 83, pp. 498-523, Apr. 1995.
- [26] K. J. Ray Liu et al. "Algorithm-Based Low power and High-Performance Multimedia Signal Processing", *Proc. of IEEE*, vol. 86, No. 6, June 1998, pp. 1155-1202.
- [27] H. K. Thapar and J. Cioffi, "A Block Processing Method for Designing High Speed Viterbi Decoders," in *Proc. IEEE Communications Conf*, June 1989, pp. 1096-1100.

- [28] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-Power CMOS Digital Design," IEEE J. Solid-State Circuits, vol. 27, PP. 473-484, Apr. 1992.
- [29] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. "Motion-compensated Interframe Coding for Video Conferencing", Proc. National Telecommun. Conf, New Orleans, LA, 1981, pp. G.5.3.1-G.5.3.5.
- [30] J. R. Jam and A. K. Jam, "Displacement Measurement and its Application in Interframe Image Coding," IEEE Trans. Commun. vol. COM-29, pp. 1799—1808, Dec. 1981.
- [31] T. H. Meng, R. Brodersen, and D. Messerschmitt, "Asynchronous Design for Programmable Digital Signal Processors. IEEE Trans. Signal Processing. vol. 39, pp. 939—952, Apr. 1991.
- [32] K. I. R. Liu, S.-F. Hsieh, and K. Yao, "Systolic Block Householder Transformation for RLS Algorithm with Two-level Pipelined Implementation," IEEE Trans. Acoust., Speech. Signal Processing, vol. 40, pp. 946-958. Apr. 1992.
- [33] C. L. Wang, "Bit-serial VLSI Implementation of Delayed LMS Adaptive FIR filter," IEEE Trans. Signal Processing, vol. 42, pp. 2169—2175, Aug. 1994.
- [34] A. V. Oppenheim and R. W. Schaffer. Discrete-Time Signal Processing. Englewood Cliffs, NJ: Prentice-Hall. 1989
- [35] I. Hong, D. Kirovski et al., "Power Optimization of Variable Voltage Core Based Systems", Proceedings of Design Automation Conference, 1998, pp. 176-181.
- [36] Y. Li, T. Callahan et al., "Hardware-Software Co-Design of Embedded Reconfigurable Architectures", Proceedings, 37th Design Automation Conference, 2000.
- [37] Z. Huang and S. Malik, "Managing Dynamic Reconfiguration Overhead in Systems on Chip Design using Reconfigurable Datapaths and Optimized Interconnection Networks", DATE 2001, March 2001.
- [38] www.ee.princeton.edu/ESRG/pleiades
- [39] R. Phelps, M. J. Krasnicki et al., "A Case Study of Synthesis for Industrial-Scale Analog IP: Re-design of the Equalizer/Filter Frontend for an ADSL CODEC", Proceedings of Design Automation Conference, 2000, pp. 1-6.
- [40] Mixed signal design, www.mentor.com/ams
- [41] Vijay K. Madiseti and Lan Shen, "Interface Design for Core based Systems", IEEE Design & Test of Computers, pp. 42-51, 1997.
- [42] www.chips.ibm.com/products/coreconnect
- [43] www.lexra.com/lx4180.html
- [44] www.sonicsinc.com
- [45] www.chronology.com
- [46] M. E. Shobaki, L. Lindh "A Hardware and Software Monitor for High level System on Chip Verification", Proc IEEE Symposium on Quality Electronic Design, 2001.
- [47] IEEE P1500, <http://grouper.ieee.org/groups/1500>
- [48] <http://www.ieee-isto.org/Nexus5001>
- [49] Mixed Signal Test Bus, www.manta.ieee.org/groups/1149/4
- [50] H. J. Wunderlich, "BIST for systems on chip", Integration, the VLSI Journal, 26 (1998) Elsevier, pp. 55-78