The Dominance, Challenges and Resolution of Giga-Scale Integration System-On-A-Chip Design

Ishtiaq Ahammad

MASTER OF SCIENCE IN INFORMATION AND COMMUNICATION ENGINEERING
NOAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

SUPERVISOR'S DECLARATION

We hereby declare that we have checked this thesis and in our opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Master of Science in Information and Communication Engineering.

_____

Name of Supervisor: DR. MD. ASHIKUR RAHMAN KHAN

Position: PROFESSOR & CHAIRMAN

Date:

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is my own except for quotations, references and summaries which have been duly acknowledged. This thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.

_____

Ishtiaq Ahammad

Roll: ASH1411048M

Date:

ACKNOWLEDGEMENTS

I am grateful and would like to express my sincere gratitude to my supervisor Professor Dr. Md. Ashikur Rahman Khan for his germinal ideas, invaluable guidance, continuous encouragement and constant support in making this research possible. He has always impressed me with his outstanding professional conduct, his strong conviction for science, and his belief that a M.Sc. program is only a start of a life-long learning experience. I appreciate his consistent support from the first day I applied to graduate program to these concluding moments. I am truly grateful for his progressive vision about my training in science, his tolerance of my native mistakes, and his commitment to my future carrier. I also would like to express very special thanks to my co-supervisor Zayed-Us-Salehin for his suggestions and co-operations throughout the study. I also sincerely thanks for the time spent proofreading and correcting my many mistakes.

My sincere thanks go to all my lab mates, members of the staff of the Information and Communication Engineering Department, NSTU, who helped me in many ways and made my stay at NSTU pleasant and unforgettable.

I acknowledge my sincere indebtedness and gratitude to my parents for their love, dream and sacrifice throughout my life. Special thanks should be given to my committee members. I would like to acknowledge their comments and suggestions which were crucial for the successful completion of this study.

Chapter 1: Introduction

This chapter at a glance:

1.1 Introduction

Giga-scale Integration (GSI) is a designation in microprocessor designs where integrated circuits (IC) contain more than one billion transistor gates. It refers to very dense proliferation of transistors on IC systems. In a practical sense, GSI is a benchmark measurement for microprocessors; it shows how far processor design has come with modern strategies like multi-core design, etc. It is part of an evolving hardware advancement that has powered tech innovation over the last few years. New microprocessor designs use a process known as photolithography to embed large amounts of circuitry on semiconductor substrates. This can facilitate GSI and related goals [7]. A system-on-a-chip(SoC) is an Integrated Circuit(IC) in which all the components needed for a computer or other system are included on a single chip. The design of such a device can be complex and costly. However these drawbacks are offset by lower manufacturing and assembly cost and by a greatly reduced power budget. Because signals among the components are kept on-die, much less power is required [8]. The size of a SoC is also its greatest disadvantages. Since all the components are compressed into a single integrated circuit, they are limited in their storage capacity and processing power. For example, a high-end desktop computer with a dedicated graphics card and SSD will easily outperform a smartphone from the same generation. However, advances in mobile processing technology enable modern smartphones to provide similar performance to high-end computers from only a few years ago [9]. The driving force behind the spectacular advancement of integrated circuit technology in the past fifty years has been the exponential scaling of the feature size, i.e., the minimum dimension of a transistor. It has been following Moore's Law at the rate of a factor of 0.7% reduction every three years. It is expected that such exponential scaling will continue for at least another 10 to 12 years as projected in the 1997 National Technology Roadmap for Semiconductors (NTRS'97). This lead to over half a billion transistors integrated on a single chip with an operating frequency of 2-3 GHz in the 70nm technology by Year 2009. However, in today the 14nm technology is available. The 14 nanometer (14 nm) semiconductor device fabrication node is the technology node following the 22 nm/(20 nm) node. The naming of this technology node as "14 nm" came from the International Technology Roadmap for Semiconductors (ITRS). One nanometer (nm) is one billionth of a meter. The 14nm refers to the "minimum feature size". Until about 2011, the node following 22nm was expected to be 16nm. The first 14 nm scale devices were shipped to consumers by Intel in 2014.

This enables system-on-a-chip (SoC) integration. However, the challenges to sustain such an exponential growth and to achieve such Giga-scale integration have shifted in a large degree from the process and manufacturing technologies to the design technologies. For example, the study by SEMATECH showed that although the level of on-chip integration, expressed in terms of the number of transistors per chip, increases at an approximate 58% annual compound growth rate, the design productivity, measured in terms of the number of transistors per person-month, grows only at a 21% annual compound rate. Such a mismatch of silicon capacity and design productivity, if not resolved timely, will seriously limit the potential of achieving high-degree

on-chip integration. Therefore, a great deal of innovation in design and test technologies is needed in order to extend Moore's Law into the next two decades [2].

With rapid feature size scaling, circuit performance is increasingly determined by the interconnects instead of devices in deep submicron (DSM) designs. Rapid scaling has introduced many challenges on interconnect performance, modeling, and reliability. Since interconnects will not be finalized until physical design is carried out, there is a strong need for more research in physical design with consideration of DSM effects. We have been building large, electronic

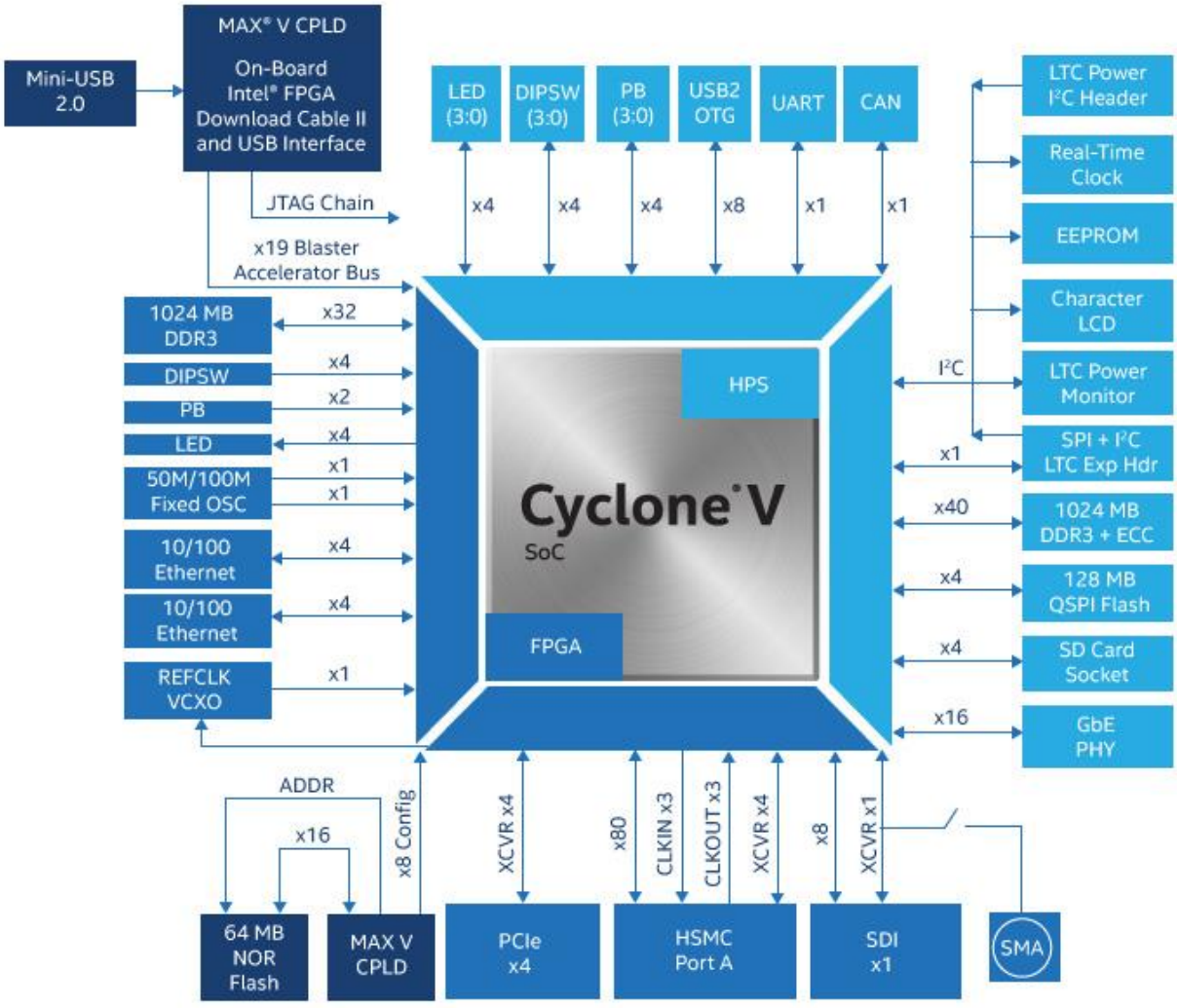

Figure 1.1: A Cyclone V SoC Architecture

systems for years, but we have only recently faced the challenge of integrating large systems on a single chip. The Giga-scale SoC era will place unprecedented demands on our system design capabilities. A SoC has been defined as the integration of computation engine, memory and logic

on a single chip. While this definition may be accurate, it understates the problems facing system designers in the Giga-scale SoC era. The challenges facing a SoC designer are more readily conveyed by a thought experiment: Consider the motherboard of a modern workstation, complete with memory, including the operating system and the applications software. Now add analog sensors and actuators and wireless communications. Future systems designers will face this level of complexity. Furthermore, the couplings and interactions among system components will increase as we put more of the system on silicon die [1].

SoC designs are highly complex, not only in terms of the high transistor count, but also in terms of large numbers of heterogeneous components and technologies to be integrated on a single chip. A Giga-scale SoC may include embedded CPUs, DSPs, FPGAs, various kinds of embedded memories, such as SRAMs, DRAMs, and FLASH memories, as well as possible multiple analog and RF components. Therefore, SoC designs require a wide range of knowledge and design experience as well as a complex set of software tools to support such heterogeneous integration throughout the entire design process. Although the promise and importance of SoC designs have been well recognized in the past few years, only limited research related to SoC designs has been reported, largely due to the fact that a single researcher or research group usually has expertise in one or a few areas related to SoC designs, and has difficulties developing a complete SoC application and comprehending the entire SoC design process.

1.2 Summary of My Thesis Work

In the future, the new generation of GSI SoC architecture must meet more open and challenging design issues. Twenty-first century opportunities for GSI will be governed in part by a hierarchy of physical limits on interconnects whose levels are codified as fundamental, material, device, circuit, and system. The International Technology Roadmap for Semiconductors (ITRS) projects that by 2011 over one billion transistors will be integrated into a single monolithic die. The wiring system of this billion-transistor die will deliver power to each transistor, provide a low-skew synchronizing clock to latches and dynamic circuits, and distribute data and control signals throughout the chip. The resulting design and modeling complexity of this GSI multilevel interconnect network is enormous such that over 10 coupling inductances and capacitances throughout a nine-to-ten-level metal stack must be managed.

Significant progress has been made in the past few decades in the logic-level and RT-level synthesis, which successfully raised the design abstraction from schematic diagrams to HDLs (hardware description languages). In order to maintain such a high level of design abstractions in the presence of increasing interconnect delays and design complexity in nanometer designs, much research is needed. Because SoC design is still at its infancy, the requirements for the design technologies are not totally clear.

Over the past ten years, as integrated circuits became increasingly more complex and expensive, the industry began to embrace new design and reuse methodologies that are collectively referred to as system-on-chip (SoC) design. We focus on the reuse and integration issues encountered in the paradigm shift. The reusable components, called intellectual property (IP) blocks or cores, are typically synthesizable register-transfer level (RTL) designs (often called soft cores) or layout level designs (often called hard cores). The concept of reuse can be carried out at the block, platform, or chip levels, and involves making the IP sufficiently general, configurable, or programmable, for use in a wide range of applications.

Today every processing system become very fast and power efficient due to fabrication of multiple system on a single chip .This multiple system need interconnect to communicate with each other through a bus. That‘s why different bus architecture was developed based on different future goal. It also imposes new requirements and challenges. System complexity increases at the same speed. Now a day the system could never be design using the same approaches applied 20 years ago. New architectures are and must be continuously conceived. System-on-chip (SoC) designers today are faced with incredible complexity in the light of billion transistor designs that have already become a reality and ever increasing numbers of components(processors, memories, peripherals, custom hardware) being integrated on a single chip. Recent advances in silicon densities now allow for the integration of numerous functions onto a single silicon chip. With this increased density, peripherals formerly attached to the processor at the card level are integrated onto the same die as the processor. As a result, chip designers must now address issues traditionally handled by the system designer. Designs such as these are best realized through a macro-based approach. Macro based design provides numerous benefits during logic entry and verification, but the ability to reuse intellectual property is often the most significant. From generic serial ports to complex memory controllers and processor cores, each SoC generally requires the use of common macros. Many single chip solutions used in applications today are designed as custom chips, each with its own internal architecture. Logical units within such a chip are often difficult to extract and re-use in different applications. As a result, many times the same function is redesigned from one application to another. Promoting reuse by ensuring macro interconnectivity is accomplished by using common buses for inter macro communications.

Verification and test are increasingly becoming bottlenecks in designing highly complex integrated systems (e.g. GSI SoC).

This work presents an overview of the challenges that face GSI SoC designers and the synthesis by which the challenges might be overcome. This work also present two emerging resolutions which can be followed when developing a GSI SoC. The required verification and testing methodologies for design is also present.

1.3 Objective of My Thesis Work

- To investigate and develop innovative design and test methodologies and tools to enable efficient Giga-scale SoC integration in nanometer technologies.
- Identify the areas of challenges related to physical design and On-chip communication.
- Find out the future challenges facing GSI SoC designers.
- Develop verification and test techniques to achieve an ultimate goal of testing and verification capabilities.
- Provide efficient synthesis environment to overcome the challenges.
- Present few emerging resolutions of design.

1.4 Organization of Thesis

This thesis consists of 9 chapters and these chapters are organized as follows:

Chapter 1: Introduction (Explain the concept of GSI SoC, its limits and work scope based on the limits, represent the objectives and summary of this thesis work).

Chapter 2: Literature review (Describes Giga-Scale Integration and System-On-a-Chip in details and shown the previous work and requirements needed).

Chapter 3: Problem Statement (Describe challenges of GSI SoC, what we are going to working with).

Chapter 4: Design Challenges (Describe physical design challenges, challenges on on-chip communication and future design challenges in details).

Chapter 5: Synthesis (Methodology) (Describe areas needed to research, design driver and synthesis environment in details).

Chapter 6: Emerging Resolutions (Describe two emerging resolution in details).

Chapter 7: Verification and Testing (Describe verification and test methodologies in details).

Chapter 8: Analytical Results and Discussion (Describe analytical results and discussion and shown the overall summation).

Chapter 9: Conclusion and future work (Describe conclusion and future work scope).

Chapter 2: Literature Review

This chapter at a glance:

2.1 What is Giga-Scale Integration (GSI)?

In computer microprocessor manufacturing, term for the design and production of integrated circuits with a component density of over 1 billion transistor gates and opportunities for incorporation of more than one billion transistors and associated interconnections within a single silicon chip stands for Giga-scale integration (GSI). It refers to very dense proliferation of transistors on IC systems. In a practical sense, GSI is a benchmark measurement for microprocessors; it shows how far processor design has come with modern strategies like multi-core design, etc. It is part of an evolving hardware advancement that has powered tech innovation over the last few years. New microprocessor designs use a process known as photolithography to embed large amounts of circuitry on semiconductor substrates. This can facilitate GSI and related goals. There are different opinions on the future of microprocessor advancement, but the use of terms like Giga-scale integration indicates there is still room for advancement in putting even more logical design into smaller and smaller chips.

2.1.1 Development of GSI

The first semiconductor chips held two transistors each. Subsequent advances added more transistors, and as a consequence, more individual functions or systems were integrated over time. The first integrated circuits held only a few devices, perhaps as many as ten diodes, transistors, resistors and capacitors, making it possible to fabricate one or more logic gates on a single device. Now known retrospectively as small-scale integration (SSI), improvements in technique led to devices with hundreds of logic gates, known as medium-scale integration (MSI). Further improvements led to large-scale integration (LSI), i.e. systems with at least a thousand logic gates. Current technology has moved far past this mark and today's microprocessor have many millions of gates and billions of individual transistors. At one time, there was an effort to name and calibrate various levels of large-scale integration above VLSI. Terms like ultra large-scale integration (ULSI) were used. But the huge number of gates and transistors available on common devices has rendered such fine distinctions moot. Terms suggesting greater than VLSI levels of integration are no longer in widespread use [8].

In 2008, billion-transistor processors became commercially available. This became more commonplace as semiconductor fabrication advanced from the then-current generation of 65 nm processes. Current designs, unlike the earliest devices, use extensive design automation and automated logic synthesis to lay out the transistors, enabling higher levels of complexity in the resulting logic functionality. Certain high-performance logic blocks like the SRAM (static random-access memory) cell, are still designed by hand to ensure the highest efficiency.

As of 2016, transistor counts continue to grow beyond ten billion transistors per chip. Multiple developments were required to achieve this increased density. Manufacturers moved to smaller design rules and cleaner fabrication facilities so that they could make chips with more transistors and maintain adequate yield. The path of process improvements was summarized by the International Technology Roadmap for Semiconductor (ITRS), which has since been succeeded by the International Roadmap for Devices and Systems (IRDS). Electronic design tools improved enough to make it practical to finish these designs in a reasonable time. The more energy-efficient CMOS replaced NMOS and PMOS, avoiding a prohibitive increase

in power consumption. Modern VLSI devices contain so many transistors, layers, interconnections, and other features that it is no longer feasible to check the masks or do the original design by hand. Instead, engineers use EDA tools to perform most functional verification work.

Name	Significance	Year	Transistor Number	Logic Gates Number
SSI	Small-scale Integration	1964	1 to 10	1 to 12
MSI	Medium-scale Integration	1968	10 to 500	13 to 99
LSI	Large-scale Integration	1971	500 to 2000	100 to 9999
VLSI	Very Large-scale Integration	1980	20000 to 1000000	10000 to 99999
ULSI	Ultra Large-scale Integration	1984	1000000 and more	100000 and more
GSI	Giga-scale Integration	1990	10000000 and more	1000000 and more

Table 2.1: Generations of Integrated Circuits

2.1.2 Moore's Law for Integrated Circuits (ICs)

Moore's Law is the observation made by Intel co-founder Gordon Moore that the number of transistors on a chip doubles every year while the costs are halved. In 1965, Gordon Moore noticed that the number of transistors per square inch on integrated circuits had doubled every year since their invention. Moore's law predicts that this trend will continue into the foreseeable future. Although the recent pace has slowed for Moore's Law, the doubling of installed transistors on silicon chips occurs closer to every 18 months instead of annually. The 18-month mark is the current definition of Moore's law. Moore's law suggests exponential growth. Thus, it is unlikely to continue indefinitely. Most experts expect Moore's law to hold for another two decades. Many experts believe Moore's Law hit its physical and economical limitations in 2017 and has slowed. The extension of Moore's law is that computers, machines that run on computers and computing power all become smaller and faster with time, as transistors on integrated circuits become more efficient. Transistors are simple electronic on/off switches embedded in microchips, processors and tiny electrical circuits. The faster microchips process electrical signals, the more efficient a computer becomes. Costs of these higher-powered computers eventually came down too, usually about 30 percent per year. When designers increased the

performance of computers with better-integrated circuits, manufacturers could create better machines that could automate certain processes. This automation created lower-priced products for consumers, as the hardware created lower labor costs [28].

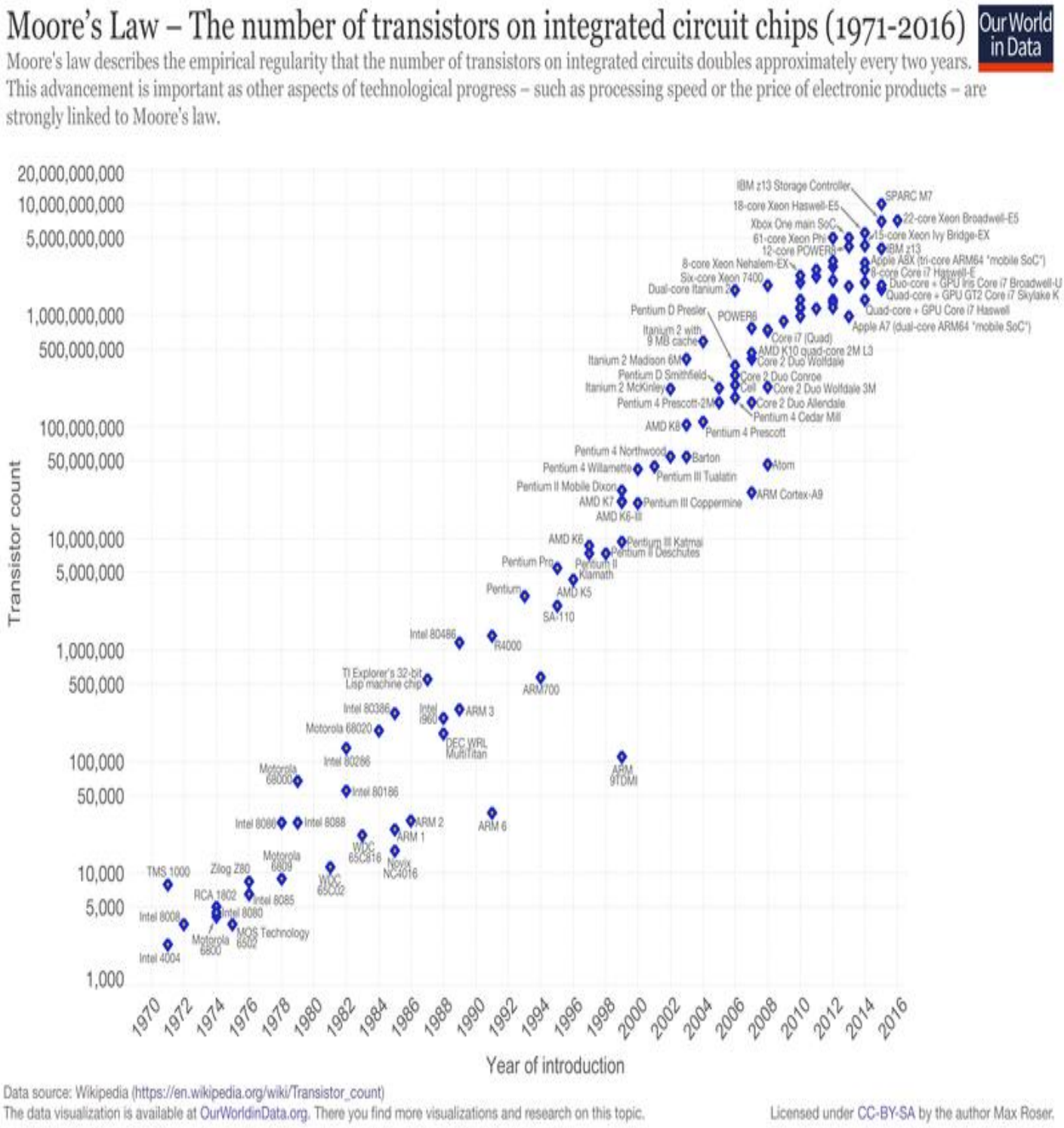

Figure 2.1: Moore's Law- The Number of Transistors on IC Chips

2.1.2.1 Contemporary Society

Fifty years after Moore's law, contemporary society sees dozens of benefits from his vision. Mobile devices, such as smartphones and tablet computers, would not work without tiny processors. Smaller and faster computers improve transportation, health care, education and energy production. Just about every facet of a high-tech society benefits from the concept of Moore's law put into practice.

2.1.2.2 The Future

Thanks to nanotechnology, some transistors are smaller than a virus. These microscopic structures contain carbon and silicon molecules aligned in perfect fashion to help move electricity along the circuit faster. Eventually, the temperature of the transistors makes it impossible to create smaller circuits, because cooling the transistors takes more energy than what passes through the transistors. Experts show that computers should reach physical limits of Moore's law in the 2020s. When that happens, computer scientists can examine entirely new ways of creating computers. Rather than physical processes, applications and software can improve the speed and efficiency of computers moving forward. Cloud Computing, wireless communication, the Internet of Things and quantum physics may all play a role in innovating computer technology. Many designers, engineers, and computer scientists agreed in early 2016 that Moore's law might run its course within 10 years. Progress achieving the doubling of the number of circuits has slowed, and integrated circuits cannot get much smaller as transistors approach the size of an atom. At some point, software or hardware breakthroughs may keep the dream of Moore's law alive. However, the computer industry seems ready to move on to another course.

2.2 What is System-On-A Chip (SoC)?

A system-on-a-chip (SoC) is a microchip with all the necessary electronic circuits and parts for a given system, such as a smartphone or wearable computer, on a single integrated circuit (IC). A SoC for a sound-detecting device, for example, might include an audio receiver, an analog-to-digital converter (ADC), a microprocessor, memory, and the input/output logic control for a user - all on a single chip. System-on-a-chip technology is used in small, increasingly complex consumer electronic devices. Some such devices have more processing power and memory than a typical 10-year-old desktop computer. In the future, SoC-equipped nanorobots (robots of microscopic dimensions) might act as programmable antibodies to fend off previously incurable diseases. SoC video devices might be embedded in the brains of blind people, allowing them to see and SoC audio devices might allow deaf people to hear. Handheld computers with small whip antennas might someday be capable of browsing the Internet at megabit-per-second speeds from any point on the surface of the earth [29].

2.2.1 Types of SoC

In general, there are four distinguishable types of SoCs:

- SoCs built around a microcontroller,
- SoCs built around a microprocessor, often found in mobile phones;
- Specialized SoCs designed for specialize application that do not fit into the above two categories, and
- Programmable systems-on-chip (PSoC), where most functionality is fixed but some functionality is reprogrammable in a manner analogous to a field-programmable gate array [8].

2.2.2 Applications of SoC

Systems-on-chip can be applied to any computing task. However, they are typically used in mobile computing such as tablets, smartphones, smartwatches and netbooks as well as embedded systems and in applications where previously microcontrollers would be used.

Mobile computing based SoCs include:

- Apple: Apple-designed processors
  - A12 Bionic and other A series, used in iPhones and iPads
  - S series and W series, in Apple Watches.
  - Apple T series, used in the 2016 and 2017 MacBook Pro fingerprint scanners.
- Samsung Electronics: List, typically based on ARM7 and ARM9
  - Exynos, used mainly by Samsung's Galaxy series of smartphones
- Qualcomm:
  - Snapdragon (list), used in many LG, Xiomi, Google Pixel, HTC and Samsung Galaxy smartphones. In 2018, Snapdragon SoCs are being used as the backbone of Laptop Computers running Windows 10, marketed as "Always Connected PCs" [8].

2.3 Previous Work

The design of a modern GSI System-on-Chip (SoC) is complex task involving a range of skills and a deep understanding of a hierarchy of perspectives on design, from processor architecture down to signal integrity. An International Workshop on Challenges and Opportunities in Giga-Scale Integration for System-On-A-Chip, jointly sponsored by the U.S. National Science Foundation (NSF) and Taiwan National Science Council (NSC), was organized by Professor Jason Cong from the University of California, Los Angeles, and Professors Youn-Long Lin and C. L. Liu from the National Tsinghua University, Taiwan. It was held in Hsin-Chu, Taiwan, during August 24 — 26, 1999. The goal of this international workshop was to understand the

challenges and critical research needs of design and test technologies for giga-scale system-on-a-chip (SoC) integration in nanometer technologies and identify promising opportunities for innovation. In the system-on-a-chip design era, system integration happens at fabrication foundries during IC manufacturing, as opposed to in system houses through PCB board-level assembly as in the past. The objective of this workshop was to understand the challenges and critical research needs of design and test technologies for giga-scale SoC integration in nanometer technologies, and identify promising opportunities for innovation. As the result of intensive discussions at the workshop, a set of important challenges and critical research needs was identified.

A thesis developed in the year 1995 by James D.Meindl and Joseph M. Pettit (Chair Professor of Microelectronics), Georgia Institute of Technology, describes what will be the 21st Century Giga-scale Integration (GSI) and what challenges will they face. Throughout the past 35 years integrated electronics has advanced at a pace unmatched in technological history. Minimum feature sizes have declined by about a factor of 1/50, switching energy of a binary transition has decreased by approximately U105 times, the number of transistors per chip has multiplied by around 50 **x** 106, the price range of a chip has remained virtually unchanged and its reliability has increased manifold. As a consequence of these unprecedented advances, integrated electronics has been the principal driver of the modern information revolution. The ubiquitous microchip has had a profound and pervasive impact on our lives. Consequently, future prospects for continuing the technological advance of integrated electronics are of paramount importance to society as a whole. The central thesis of this discussion was that future opportunities for multi-billion transistor chips - described as giga-scale integration (GSI) - in the 21st century will be governed by a hierarchy of physical limits. The levels of this hierarchy can be codified as 1) fundamental, 2) material, 3) device, 4) circuit and *5)* system.

Three key fundamental limits are derived from thermodynamics, quantum mechanics and electromagnetic theory. The most important material limits are imposed by carrier mobility, saturation velocity and velocity overshoot, as well as the breakdown field strength and thermal conductivity of silicon and gallium arsenide. The critical device limits are concerned with minimum channel length, gate insulator thickness and junction depth in field effect transistors. Four generic circuit limits are imposed by the transfer characteristic of a logic circuit, its power delay product, its intrinsic speed and the response time of global interconnects. System limits are the most restrictive ones in the hierarchy and are based upon chip architecture, the power-delay product of the associated semiconductor technology, the heat removal and interconnect capacity of the packaging technology, timing requirements and overall physical size. A distinctive benefit of establishing a coherent hierarchy of limits on GSI is that it provides a constructive context which prompts revealing questions, enriches understanding and reinforces conclusions regarding promising prospects for approaching and circumventing the most important limits.

A thesis developed by Jeffery A. Davis, Raghuraman Venkatesan, Alain Kaloyeros, Michael beylansky, Shukri J. Souri, Kaustav BanerjeeE, Member, IEEE, Krishna C. Saraswat, Fellow,

IEEE, Arifur Rahman, Member, IEEE, Rafael Reif, Fellow, IEEE, and James D. Meindl, Fellow, IEEE, describes the Interconnect Limits on Giga-scale Integration (GSI) in the 21st Century. Twenty-first century opportunities for GSI will be governed in part by a hierarchy of physical limits on interconnects whose levels are codified as fundamental, material, device, circuit, and system. Fundamental limits are derived from the basic axioms of electromagnetic, communication, and thermodynamic theories, which immutably restrict interconnect performance, energy dissipation, and noise reduction. At the material level, the conductor resistivity increases substantially in sub-50-nm technology due to scattering mechanisms that are controlled by quantum mechanical phenomena and structural/morphological effects. At the device and circuit level, interconnect scaling significantly increases interconnect crosstalk and latency. Reverse scaling of global interconnects causes inductance to influence on-chip interconnect transients such that even with ideal return paths, mutual inductance increases crosstalk by up to 60% over that predicted by conventional RC models. At the system level, the number of metal levels explodes for highly connected 2-D logic mega cells that double in size every two years such that by 2014 the number is significantly larger than ITRS projections. This result emphasizes that changes in design, technology, and architecture are needed to cope with the onslaught of wiring demands. One potential solution is 3-D integration of transistors, which is expected to significantly improve interconnect performance. Increasing the number of active layers, including the use of separate layers for repeaters, and optimizing the wiring network, yields an improvement in interconnect performance of up to 145% at the 50-nm node.

2.4 Requirements Needed

- Design Driver (Network Processor)
- C-VHDL multi-language specification
- C-VHDL co-simulator
- Algorithms/Methods (to partition the C-VHDL-based system specification into hardware/software subsystems)
- JBRC (Jade Bird Retargetable Compiler)
- JBISASim (trace-driven instruction-set architecture evaluation tools)
- JBCacheSim (cache performance evaluation tools)
- JBWorkBench (programmer's workbench)
- RTL/JBASM-SL Based Instruction Level Simulator

Chapter 3: Problem Statement

Problem Statement

The objective of this work is to find out the physical challenge areas and the future challenges that might face GSI SoC designers and provide synthesis and emerging resolutions to overcome these challenges. Due to shortage of time we only able to present two emerging resolutions but by following the synthesis environment that we present, one can easily find out more solution and can understand what design he/she would follow. This analysis is to be carried out through exhaustive literature review, design challenges and synthesis.

The general objectives can be outlined as follows:

- Get a general understanding of GSI SoC design
- Literature review of GSI SoC and design challenge areas
- Provide physical design challenges and future challenges
- Present synthesis environment to overcome challenges
- Present two emerging resolutions that carried out the most effective design
- Analyze the result of the proposed work

The areas needed to be concerned and challenges facing Giga-scale SoC designers can be divided into the following areas:

• **Design methodology:** We need to design at much higher levels of abstraction.
• **Core-based design:** Much of our design work will be in the design and utilization of reusable cores.
• **Intellectual property issues:** Once the cores have been designed, there are many challenges in importing, exporting, documenting and reusing these cores.
• **Architectural design issues:** We need much better tools and methodologies to support architectural design.
• **Multi-disciplinary design:** Future system designers need to make tradeoffs over a broader range of issues because of the closer interactions required by SoC.
• **Increased productivity:** The need for increased productivity underlies all of the other areas, but we have devoted a separate section to productivity.
• **Miscellaneous challenges:** Some important issues did not fit into broader categories. However, we feel that they are important, so we have grouped them under miscellaneous.

Chapter 4: Design Challenges

This chapter at a glance:

4.1 Physical Design Challenges

4.2 Design Challenges for On-Chip Communication Architectures

- 4.2.1 Technology issues
- 4.2.2 Performance issues
- 4.2.3 Design Productivity issues

4.3 Future System Design Challenges

- 4.3.1 Scalable or Reusable Architecture
- 4.3.2 IP Integration
- 4.3.3 Network-on-Chip
- 4.3.4 Embedded Memory
- 4.3.5 Reliability

4.1 Physical Design Challenges

With rapid feature size scaling, circuit performance is increasingly determined by the inter connects instead of devices in deep submicron (DSM) designs. Rapid scaling has introduced many challenges on interconnect performance, modeling, and reliability. Since interconnects will not be finalized until physical design is carried out, there is a strong need for more research in physical design with consideration of DSM effects. In particular, the following areas related to physical designs are identified as crucial for system-on-a-chip integration [1].

• **Timing closure** is the key problem facing today's designers of high-speed GSI circuits and systems. Timing closure can be obtained not only through development of parasitic-aware, constraint-driven design tools, but also through introduction and adoption of integrated design methodologies and flows. Layout-driven synthesis and synthesis-driven layout are two common ways to approach this problem. Integrated CAD databases, well characterized cells, and more powerful techniques for concurrent placement/routing and logic optimization are required to achieve timing closure in a cell-based ASIC design methodology.
• **Power closure** is becoming an important design concern in some applications, including battery-powered microelectronics where maximizing the battery service life is one primary driver, and high-end systems where excessive power dissipation limits the number of devices that can be integrated on the same chip. Minimization of switched capacitances during physical design and development of tools and methodologies that support multiple Vdd and multiple Vt or even dynamically varying Vdd and clock frequency in response to varying computational workloads are important steps in achieving power closure. Thermal management is a closely related problem that also needs attention.
• **Power/ground network design, clock distribution, and signal integrity-related problems** are of primary concern in DSM designs. The former includes the IR drop and the bounce problem, whereas the latter includes signal cross talk, noise coupling through substrate, and mixed-signal interference. These problems should be considered, especially in view of the process technology and temperature variations. There is, therefore, a need to develop statistical modeling and analysis tools as well as physical synthesis techniques that are aware of such variations and optimize the circuit accordingly.
• **Integrated package-chip modeling, co-analysis and co-design** are required to provide the accuracy needed for DSM designs. Package design should be considered during IC chip planning.
• **Process migration** is required in many instances where we must retarget an existing design cost-effectively and painlessly to a new process technology or a new cell library.
• **Reliability** must be designed from the early design phase and not as a backend process. This includes, for example, considerations for electro-migration, I/O protection circuitry, latch-up, hot carrier effects, and so on.
• **Better characterization of small cells and IP blocks** must be provided for power, timing and P/G noise so as to allow more effective design tradeoffs.
• **Performance and reliability-aware design rules** are to be developed to help manage and/or overcome potential signal integrity, power, substrate noise, and reliability problems without having to rely solely on simulation.
• **On-chip optical interconnects** look promising in view of their performance, noise immunity and area comparison with respect to the conventional metallic interconnects. Wave division

multiplexing has become a mature technology in opto-electronics and may be applied for on-chip interconnects.
• **Mixed-signal design** issues are critical in many applications. Improved techniques are needed for noise shielding and guard-banding.
• **On-the-fly extraction and order reduction** of parasitic RLC and parasitic device is required to handle the complexities of the DSM designs.
• **New component models and CAD tools for MEMS and MOEMS** are needed to allow integration of mechanical and optical components on chips.
• **Computationally efficient, highly scalable algorithms for physical design** optimization are needed so that they can far extend the capabilities of the current algorithms in handling complex designs and in achieving global optimization.
• **Integrated and flexible physical design tools that consider realistic operating conditions** (temperature variations, process variations, etc.) are needed. Uniform temperature assumption may hide functional failures during simulation.

4.2 Design Challenges for On-Chip Communication Architectures

Designing communication architectures for highly integrated deep sub-micron SoCs is a non-trivial task that needs to take into account the main challenges posed by technology scaling and by exponentially increasing system complexity. A few relevant SoC design issues are hereafter discussed [3]:

4.2.1 Technology issues

While gate delays scale down with technology, global wire delays typically increase or remain constant as repeaters are inserted. It is estimated that in 50 nm technology, at a clock frequency of 10 GHz, a global wire delay can be up to 6–10 clock cycles [4]. Therefore, limiting the on-chip distance travelled by critical signals will be key to guarantee the performance of the overall system, and will be a common design guideline for all kinds of system interconnects. Synchronization of cores on future SoCs will be unfeasible due to deep submicron effects (clock skew, power associated with clock distribution trees, etc.), and an alternative scenario consists of self synchronous cores that communicate with one another through a network-centric architecture. Finally, signal integrity issues (cross-talk, power supply noise, soft errors, etc.) will lead to more transient and permanent failures of signals, logic values, devices and interconnects, thus raising the reliability concern for on-chip communication. In many cases, on-chip networks can be designed as regular structures, allowing electrical parameters of wires to be optimized and well controlled. This leads to lower communication failure probabilities, thus enabling the use of low swing signaling techniques, and to the capability of exploiting performance optimization techniques such as wave-front pipelining.

4.2.2 Performance issues

In traditional busses, all communication actors share the same bandwidth. As a consequence, performance does not scale with the level of system integration, but degrades significantly. Though, once the bus is granted to a master, access occurs with no additional delay. On the contrary, SoCs can provide much better performance scalability. No delays are experienced for accessing the communication infrastructure, since multiple outstanding transactions originated by multiple cores can be handled at the same time, resulting in a more efficient network resource utilization [5]. However, given a certain network dimension (e.g., number of instantiated switches), large latency fluctuations for packet delivery could be experienced as a consequence of network congestion. This is unacceptable when hard real time constraints of an application have to be met, and two solutions are viable: network over dimensioning (for SoCs designed to support best-effort traffic only) or implementation of dedicated mechanisms to provide guarantees for timing constrained traffic (e.g., loss-less data transport, minimal bandwidth, bounded latency, minimal throughput, etc.).

4.2.3 Design Productivity issues

It is well known that synthesis and compiler technology development does not keep up with IC manufacturing technology development [10]. Moreover, time-to-market needs to be kept as low as possible. The reuse of complex pre-verified design blocks is efficient means to increase productivity, and regards both computation resources and the communication infrastructure [11]. It would be highly desirable to have processing elements that could be employed in different platforms by means of a plug-and-play design style. To this purpose, a scalable and modular on-chip network represents a more efficient communication infrastructure compared with shared bus-based architectures. However, the reuse of processing elements is facilitated by the definition of standard network interfaces, which also make the modularity property of the SoC effective. The Virtual Socket Interface Alliance (VSIA) has attempted to set the characteristics of this interface industry-wide. OCP is another example of standard interface sockets for cores. It is worth remarking that such network interfaces also decouple the development of new cores from the evolution of new communication architectures. Finally, let us observe that SoC components (e.g., switches or interfaces) can be instantiated multiple times in the same design (as opposed to the arbiter of traditional shared busses, which is instance-specific) and reused in a large number of products targeting a specific application domain.

4.3 Future System Design Challenges

In the future, the new generation of GSI SoC architecture must meet more open and challenging design issues, as exemplified below [30]:

4.3.1 Scalable or Reusable Architecture

As multimedia applications grow in complexity, we need more and more computational capability. For example, video coding standards have been evolved from MPEG-2 to H.264 and the picture resolutions have been increased from DVD (720x480) to HDTV (1280x720 or 1920x1080). To ensure fast design turnaround time without completely redesigning the whole system, a scalable architecture is highly desired. Just as we shifted our IC design paradigm from full custom design to standard cells, even to IP reuse, the next-generation system design paradigm shift should be the reuse of architecture. That is, to cope with the growing complexity of SoCs, IP reuse may not be enough. Reuse must happen at a much higher level than it used to, e.g., architecture reuse. However, creating a design that can be efficiently reused requires a great deal of effort.

4.3.2 IP Integration

Integration requires more than simply placing components spatially together on a single chip. A few issues, for example, are outlined below:

- How to integrate analog IP safely; in particular, how to deal with noise from the analog domain to the digital domain or vice versa.
- How to deal with black-box IP.
- How to handle its I/O requests in a timely manner without over-provisioning resources for it.
- How to migrate IPs from one process technology to the next one as quickly as possible. Is synthesizable soft IP better than customized hard IP even though customized hard IP may be more efficient?
- How to effectively test and verify the whole system when the IPs come from different sources.

These are the challenges beyond simply placing components together. We need good strategies for the integration of hardware and software IP components.

4.3.3 Network-on-Chip

The performance of next-generation SoCs will be limited by the ability to efficiently connect the functional blocks together, and to accommodate their communication requirements. As applications require more computational power in the next few years, it is clear that we need multiple processors, processing elements, and functional units. Hence, communication becomes the critical path in the system. The need for synergy of processing elements and interconnect architectures becomes critical as we face tighter power budgets in the pursuit of performance targets. First, the delay and energy of communication wires do not scale down linearly as the transistor size shrinks in the future; the interconnect will consume a significant amount of chip energy. Second, because processor and interconnect architectures share a common power and thermal budget, optimizing the power consumption of each in isolation can have a significant impact on the other's power and performance. Furthermore, the unreliability of deep submicron

on-chip wire communication stimulates more challenges. While many communication architectures make use of buses or crossbars, there are some recent proposals on network-on-chip (NoC) architectures. One approach is to employ a packet-switched interconnect. The concept is similar to traditional large-scale wide-area networks, but in this case, on-chip router-based networks are used. Programmable cores access the network via packet-switched interfaces and have their packets forwarded to their destinations through a multiple-hop routing path. Nevertheless, the on-die communication architectures cannot just mimic the off-die communication. This is because the cost tradeoff of on-chip communication is quite different from that of on-chip communication. We need different designs. For example, while compression of contents in saved power consumption for off-die bus communication, compression may need more power for on-die and short-distance communication. Yet another example is shown in. Because of communication localities and the latencies, circuit-switched NoC is sometimes more attractive than the traditional packet-switched NoC.

4.3.4 Embedded Memory

As external memory bandwidth becomes a major bottleneck in the future, more on-die high-speed memory, such as cache or local buffer, will be deployed. Sometimes, embedded memory (SRAM, DRAM, flash, ROM) will be integrated onto the chip. The amount of memory integrated into an ASIC-type SoC design has increased from 20% in 1999 to 90% in 2018. Challenges arise when trying to balance efficiency and power. SRAM provides high performance, while flash memory is the best solution in terms of power consumption. The amount and the placement of each kind of memory in the SoC will greatly affect access efficiency and power. Additionally, cache may introduce indeterminate delay, cache coherence, and memory consistency challenges. Therefore, the unpredictable latencies associated with caches must be carefully accounted for. An alternative solution is to use a software-control local buffer instead of cache. A famous commercial example of this is the Cell architecture developed by Sony, Toshiba, and IBM. A software control local buffer reduces the uncertainty when it comes to latency, but of its use makes the software more complex. Furthermore, because digital, mixed-signal, RF, and memory blocks are tightly integrated, the power and substrate noise may cause sensitive blocks to suffer from functional failures. Because of the unique characteristics of the memory circuit and layout, the power and substrate noise may cause functional failures. Making embedded memories noise tolerant will be vital to a successful SoC design. 3D stacking memory is an alternative exciting technology to increase bandwidth substantially. However, despite its promising advantages, the thermal issue of multiple dies stacking together is of serious concern. In short, a high performance and low-power SoC architecture must balance the tradeoffs from various memory options.

4.3.5 Reliability

Reliability is likely to be a major focus of research as we approach the end of the decade. As future transistor sizes become smaller than 20nm, we are likely to see increasing variability in the behavior of the transistors. Smaller feature sizes lead to more failures over time from electrostatic overstressing and electro-migration. While transistors might fail, the entire system

cannot fault. We must explore mechanisms to compensate for this underlying variability in transistor behavior. To address these challenges, research is being carried out from fabrication to software. Some of the basic fault tolerance principals that we used in the era of mainframe computers can potentially be applied to today's designs. For example, we can detect when a circuit fails and shift the work to another circuit through hardware- /firmware-based self-management. Yet another possible solution is to use statistical computing techniques, which use probabilistic models to derive reliable results from unreliable components. Nonetheless, none of these techniques is a clear answer on how to efficiently address reliability issues. The bottom line is that there needs to be an architectural solution.

Chapter 5: Synthesis (Methodology)

This chapter at a glance:

5.1 Areas Needed to Research Related to the Synthesis

Significant progress has been made in the past few decades in the logic-level and RT-level synthesis, which successfully raised the design abstraction from schematic diagrams to HDLs (hardware description languages). In order to maintain such a high level of design abstractions in the presence of increasing interconnect delays and design complexity in nanometer designs, much research is needed in the following areas related to the synthesis technologies [1].

• **Synthesis to support IP creation:** Synthesis tools are needed to generate easy-to-use IP blocks for SoC designs. In SoC design, parameterized modules such as application-specific instruction-set processors (ASIPs), signal processing functional macros, application-specific memory modules, built-in self-test (BIST) logic, etc., will be widely used. Advanced synthesis tools should be able to generate IP modules that are not only reusable but also customizable by the users. Accompanying these IPs, the synthesis tools should be able to generate complete test vectors and interface information.
• **Hierarchical synthesis:** Synthesis for the whole chip is a computational intensive process. An effective hierarchical approach that can handle area, timing, and power constraints is the key to making SoC synthesis feasible. At the same time, incorporation of IPs into the synthesis process is also necessary as more and more IPs become easily accessible. The use of hierarchical synthesis makes incremental change more efficient and controllable because it can be done on local modules.
• **Low power synthesis:** Design tools are required to achieve low power dissipation in VLSI circuits. Although some commercial tools address power optimization during technology mapping and gate sizing, they do not address power optimization during such steps as technology-independent logic synthesis, state encoding, scheduling and resource binding, bus architecture design, and so on. Furthermore, the quality of commercially available automated clock gating tools is poor since they do not use sophisticated data flow graph analysis and make use of statistics of the applied data input to improve the expected power efficiency. Tools and design methodologies that support multiple Vdd and continuously varying Vdd are also lacking.
• **Layout-based synthesis:** It is more and more apparent that interconnect delay and signal integrity will dominate the performance of the design. In order to obtain high performance and accurate design, such physical design effects must be considered during synthesis.
• **Analog synthesis:** Designing or incorporating an analog block plays a critical role in heterogeneous SoC designs. The designer needs an interactive synthesis environment to explore possible architectures for the desired performance. The synthesizer should employ reduced models for efficient optimization runs during synthesis, and provide relevant models (such as timing, power, and layout) for subsequent design steps to use. Note that since analog circuits are very sensitive to noises generated from other parts of the chip, the synthesizer also needs to provide second-order effect information of the synthesized layout for the designer to consider the trade-off of the whole chip performance and the manufacturing yield.
• **Bus and interface synthesis:** It has been long recognized that interface issues are of special importance for complex design, and in particular for SoC. Bus and interconnect often dominate delay, area, power, and other metrics of SoC designs. There is a wide spectrum of important issues, and related challenges and opportunities in bus and interface related problems, including bus protocol specification, bus architecture and implementation, physical interconnect-based bus design, bus performance estimation and evaluation, flexible and fast dedicated interface design,

bus and interface wrappers, time-segmented scheduled bus design, and support for bus-related OS and compiler issues.

5.2 Design Drivers

Because SoC design is still at its infancy, the requirements for the design technologies are not totally clear. Therefore, it is important to develop a design driver, which will be a realistic design using the SoC technology, and use it to identify detailed requirements of various design technologies and provide a necessary test case to the researchers for design tool and methodology research. Therefore, the first task of our SoC synthesis is to identify a design driver. We anticipate that a network processor is chosen [33].

5.2.1 Network Processor (NP)

Software-based routers will be phased out in the near future. In order to provide the performance required by next - generation networks, specialized network processors (NPs) are needed. Their typical functions may include one or more of the following: 1) segmentation (frame) assembly and reassembly, 2) protocol recognition and classification, 3) queuing and access control, 4) traffic shaping and engineering, and 5) quality of service. The most important factors in an NP include programmability, performance, management, and routing. First, the NP must be easily programmable in order to support customization and the rapid integration of new and existing technologies. The second requirement for NP is to support a large number of high-bandwidth connections, multiple protocols, and advanced features. Also, it should be able to provide wire-speed, non-blocking performance. Support for services such as security management and enforcement, performance and traffic-flow statistics, traffic filtering, etc., is also typically required. Finally, after identifying, classifying, and accounting for incoming traffic flows, the NP should make forwarding decisions based on pre-programmed information. Given these requirements, we shall propose an advanced NP architecture as shown in Figure 5.1.
The NP will contain the following cores: CPU, DSP, memories (including ROM, SRAM, DRAM, and possibly flash memory), FPGA, IO and user-defined function (UDF), etc. These cores will serve as the drivers for the technologies to be developed in this work. They are explained below [2].

CPU	DSP
ROM	FPGA
SRAM	IO
DRAM	Flash

Figure 5.1: Network Processor Architecture

5.2.1.1 CPU

By the end of 1999, the researchers had developed an infrastructure (named JBCODES) to support microprocessor design and corresponding compiler and assembler development. JBCODES includes: JBRC (retargetable compiler), JBASM (assembler generator), JBDisASM (disassembler development kit), JBISASim (trace-driven instruction-set architecture evaluation tools), JBCacheSim (cache performance evaluation tools), and JBWorkBench (programmer's workbench). Using JBCODES, researchers has designed a 16-bit low-power embedded and compiler-friendly microprocessor core (JBCore16). The JBCore16 architecture is based on RISC principles. There are 87 instructions in its instruction set, and 3-stage pipelining is employed so that all parts of the processing and memory systems can operate continuously. Harvard architecture makes Jbcore16's pipeline more efficient. JBCore16's high code density and low-power design can satisfy the requirements of embedded systems. In addition, JBCore16 supports multiple modes of operation: user, supervisor, interrupt, fast-interrupt, and user-defined instruction. A prototype of JBCore16 was implemented on an Altera Flex FPGA, and all C programs of SPEC-92 can run efficiently on the prototype board at 20 MHz clock frequency. A 32-bit ARM-like embedded microprocessor, named JBCore32, is being developed. JBCore32 supports both the 32-bit instructions set and a corresponding compressed 16-bit instruction set, allowing the user to trade off between high performance and high code density. JBCore32 is implemented using 5-stage pipelining consisting of instruction fetch, decode, execution, memory access and write-back stages. The JBCore32 has on-chip instruction and data caches. In addition, its MMU functions could support a full demand-paged virtual memory operating system and real-time embedded operating systems. In order to make the instruction pipeline running with least structural hazards, multiple functional units are provided. The JBCORE32 is expected to run all C programs inSPEC-95 efficiently. To satisfy SoC requirements, JBCore32's on-chip bus architecture will support peripheral design reuse and efficient production test. The proposed network processor will include not only a high-performance CPU, but also some protocol engines and special-purpose processors. This will lead further refinement of JBCore32's architecture to meet the NP application.

5.2.1.2 DSP

Among all IPs, embedded processors are among the most important. For network applications, embeddable DSP core is a must. Since the target application is networking, we shall first analyze the characteristics of such an application. The design of the instruction set will emphasize code density, power consumption, performance, and compiler friendliness. The implementation will be synthesizable RTL in Verilog HDL. We shall target it for a 0.18-0.15um CMOS low-power library. Our goal is 1GOPS at no more than 500mW.We shall focus on ISA design and FPGA prototyping: 1) collect and analyze application programs in networking and communication; 2) study the characteristics of other DSP and embedded RISC including TI, ADI, Lucent, Motorola, 3DSP, ZDSP, DSP-G, BOPS, ARM, MIPS, Tensilica and ARC; 3) define the ISA with adequate consideration of code density and parallelism; 4) evaluate datapath options including multiple MAC units, on-chip memory, address generation units, SIMD support, zero loop overhead,

arithmetic saturation, power management, etc.; 5) implement the DSP using Verilog HDL and FPGA.

5.2.1.3 Embedded Memories

Embedded memories are the most popular cores in SoC applications. Specifically, embedded ROM and SRAM cores have been widely used in the industry, and embedded flash memory cores are finding more and more applications. As for embedded DRAM (eDRAM), it provides high-capacity storage at a higher data rate as compared with commodity DRAM whose data rate is limited by the number of pins available. Also, eDRAM reduces overall power consumption and hardware cost. As we enter the deep submicron age and continue to shrink the feature size, we shall find chips to be more and more pad-limited. The proposed NP without eDRAM will be such a case. Therefore, merging DRAM and logic is expected to benefit the system IC industry. In fact, the merge has begun to appear in various ASIC and microprocessor designs and advanced computer architectures. Of course there are challenges in merging DRAM and logic, such as process optimization for the combined DRAM and logic, and novel design and test methodologies for guaranteeing its performance, quality, and reliability. Testing the eDRAM is more difficult than testing the commodity DRAM. The first test issue is accessibility. Accessing the DRAM core from an external memory tester is costly when the DRAM core is embedded in a chip and surrounded by logic blocks. Proper design-for-testability (DFT) methodology must be provided for core isolation and tester access, and a price has to be paid for the hardware overhead, performance penalty, and noise and parasitic effects. Even if this is manageable, memory testers for full qualification and testing of the eDRAM will be much more expensive due to increased speed and I/O data width, and if we also consider engineering change, the overall investment will be even higher. A promising solution to this dilemma is built-in self-test (BIST). With BIST, the tester requirement for eDRAM can be minimized, and the memory tester time can be greatly reduced throughout the entire test flow of the eDRAM. Also, the total test time can be reduced since parallel testing at the memory bank as well as the chip levels is easier. We shall also investigate the built-in selfrepair technology to further eliminate the requirement of laser repair for the DRAM core. We shall investigate all necessary test, diagnosis, and repair technologies for embedded memories, including ROM, SRAM, DRAM, and flash memory.

5.2.1.4 Embedded FPGAs

Many research results and industrial forecasts have concluded that future SoC products will include embedded microprocessors, embedded memories, and customized ASICs, as well as a significant portion of embedded FPGAs. Since the design complexity is much higher and the design cycle is considerably longer for GSI SoC designs, having embedded FPGAs in a SoC solution enables the use of one product to serve multiple applications and continuous adaptation of the product to the evolving interfaces and standards without going through the lengthy design process. This will considerably reduce the time-to-market and design cost. Moreover, since the mask cost has increased sharply for nanometer designs, field-programmable SoCs (FPSoCs) will significantly reduce the manufacturing cost, as it can be easily adapted to multiple applications and/or design changes without going through refabrication. Furthermore, FPSoC provides a solution with much higher performance and smaller area and power dissipation compared to an

all-software implementation using microprocessors. We plan to use the embedded FPGA in this NP design driver and provide customized circuitry to support various kinds of routing protocols and network functions (such as various priority schemes, data compression and decompression algorithms, encryption and decryption algorithms) so that a single SoC implementation of the NP design can be reconfigurated for different network environments and performance/bandwidth/power requirements. We shall investigate several possible approaches for developing embedded FPGAs in SoC. One approach is to develop a module generator for embedded FPGAs to be used in our SoC prototype implementation. In particular, we plan to investigate the possibility of using the novel FPGA architecture recently developed at UCLA, which is based on *k*-input single-output PLA -like cells, or, *k/m*-macrocells. Each cell in this architecture can implement a single output function of up to *k* inputs and up to *m* product terms. Our study shows that a *k/m*-macrocell with only a relatively small number of product terms (*m ? k+3*) is practically equivalent to a *k*-LUT (lookup-table) in terms of functional implementation capabilities. As a result, we can afford to use *k/m*-macrocells with large *k* as basic programmable cells, which significantly reduces the level of the circuit (and the interconnects associated with each level) and achieves better circuit performance. Our experimental result indeed shows *k/m*-macrocell based FPGAs can outperform 4-LUT based FPGAs in terms of both delay and area after placement and routing. We have developed near-optimal depth-mapping algorithms for the *k/m*-macrocell based FPGA architecture. The other alternative to developing a new embedded FPGA architecture is to license an existing FPGA architecture from one of the FPGA vendors and include it in our SoC prototype. The RASP FPGA synthesis system RASP developed at UCLA can be targeted to most of the architectures from these FPGA vendors, and many algorithms in the RASP system have been adopted in the commercial synthesis systems developed by these vendors.

5.3 Synthesis Environment

Developing an automatic synthesis environment and methodology for GSI SoC designs is a highly complex and challenging task. Using the design driver described in Section 5.2, we shall explore a number of important research topics related to SoC designs, including design specification, design partitioning, retargetable compiler for embedded processors, synthesis and optimization techniques for DSPs, synthesis and technology mapping for embedded FPGAs, and a set of issues related to embedded ASIC designs. These tasks are inter-related and form an exploratory SoC design environment as shown in Figure 5.2.

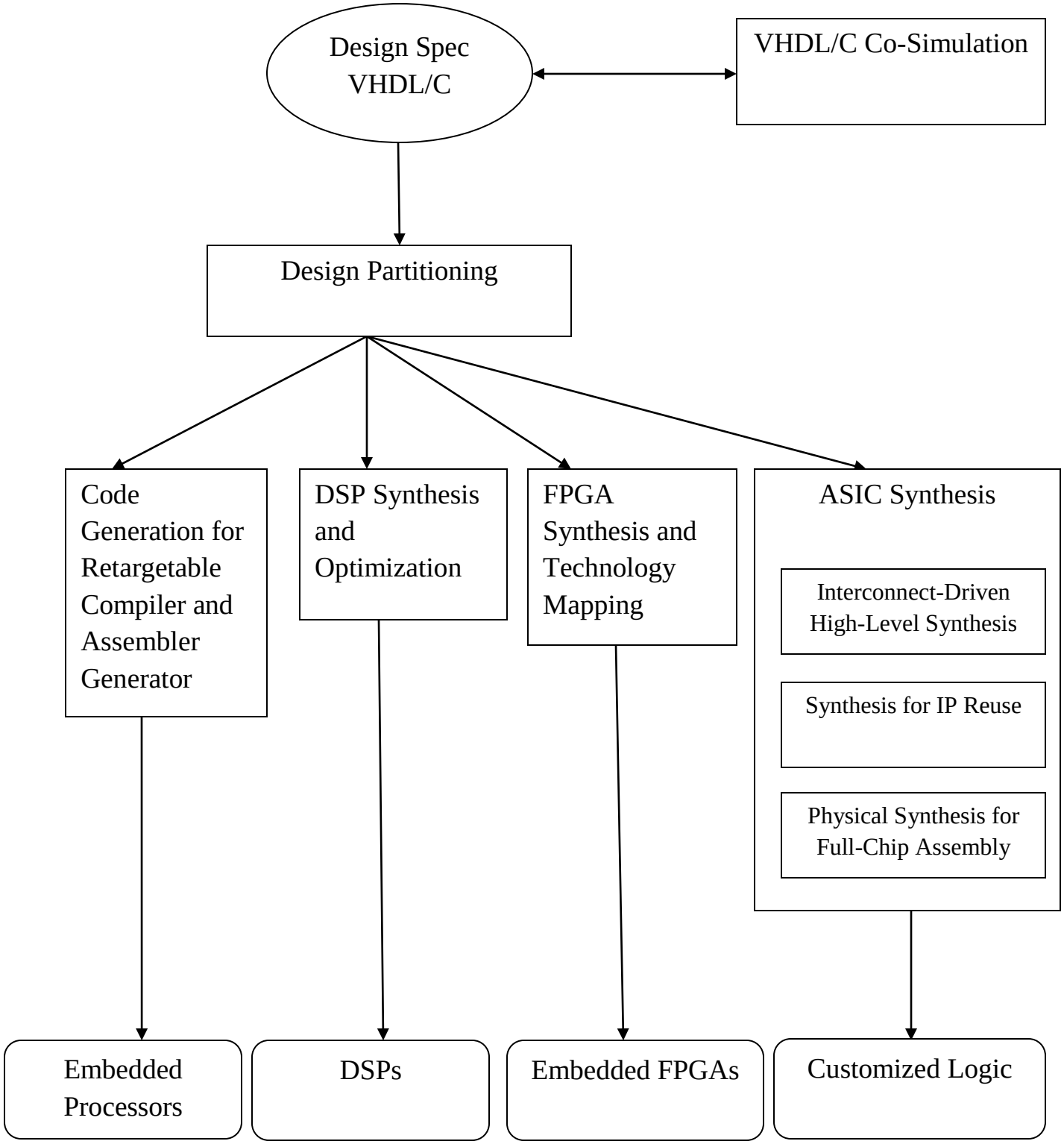

Figure 5.2: Overview of SoC Synthesis Environment

5.3.1 Design Specification

An SoC can implement a complete system including processors, memory, control circuits, programmable logic, and even analog and mixed signal circuits on a single chip.Given the complexity of SoC designs, the designer needs efficient ways to specify the system-level design and a convenient design environment to help develop SoC chips. We plan to provide such a hardware and software co-specification and co-design environment for SoC. Several co-design systems or tools have been developed in recent years. Some of them use standard C or extended C to describe systems; some of them use special languages such as Esterel; while SLS of TIMA uses multi-languages to describe the individual modules. The key issue with such co-design environments is the correspondence between the presentations used in the initial specification and the function provided by the target model (virtual prototype). For instance, the mapping of the system specification language, including high-level concepts such as distributed control and abstract communication, onto low-level languages such as C and VHDL is a non-trivial task [34]. In this paper, we propose to develop an integrated hardware/software co-design environment for SoC. It includes a multi-language system description using C and VHDL, with C-VHDL co-simulation and co-verification. Our research will include the following tasks.

(1) C-VHDL multi-language specification: In multi-language specification, one can use different languages (C, C++, or VHDL) for different modules. It allows the user to describe a system using a combination of C or VHDL. The software modules are described in C or C++ in general, while the hardware modules can be described in both C and VHDL. In this approach, the key challenges are multi-language co-verification and interface synthesis.

(2) C to VHDL transformation: After HW/SW partitioning, the C description of hardware parts will be transformed to a VHDL description. The key problem is to find the corresponding relationship between C and VHDL.

(3) C-VHDL co-simulation: A VHDL simulator prototype which transforms VHDL to C++ and runs the C++ program to realize the simulation. We only need to merge the transformed C++ programs and C specification as one executable program to realize co-simulation. The key problem is to determine how they interact and communicate with each other.

(4) C-VHDL co-verification: A work has been done on formal verification for VHDL synchronous design using a model-checking technique. We shall expand it to handle C-VHDL heterogeneous specification. Finite State Machine will be extracted from the C description. Then, we can apply the existing approach to verify it. In order to cope with complex relationship between various modules, we plan to use a parallel-hierarchical FSM model to represent the system.

5.3.2 Design Partitioning

SoC-based systems are usually implemented with mixed hardware and software components. The functionality of a system can be realized either as software components running on a processor core or application-specific hardware components. Translating a system specification into a target system architecture is referred to as the hardware/software partitioning problem. Since system-design specification is done using mixed the C and VHDL languages(as described in Section 5.3.1), in this paper, we now describe how to convert the C-VHDL system specification into the target SoC architecture. We can study the following two topics:

(1) In order to support system partitioning design decisions, we need to develop a series of C-VHDL-based design estimation and exploration techniques. We shall develop cost/performance estimation methods from the C-VHDL based system specification. Based on the estimation methods, we shall develop design exploration techniques by trading-off hardware/software implementation of different parts of the system.
(2) We shall develop algorithms/methods to partition the C-VHDL-based system specification into hardware/software subsystems. If we decide to implement part of the C specification using a hardware approach, we shall use the C to VHDL transformation method (developed in the research of Section 5.3.1) to convert the C specification into a VHDL-based hardware specification. Finally, we shall use the C-VHDL co-simulation and co-verification methods (section 5.3.2) to verify the partitioning result.

5.3.3 Interconnect-Driven High-Level Synthesis

As the feature size of the integrated circuits is down to deep sub-micron, or even nanometer technologies, traditional methods for high-level synthesis without considering the effect of interconnections are no longer able to meet the high performance requirement as interconnect delay becomes the dominating factor in determining the system performance. In this work, we propose research and development of an interconnection-oriented high-level synthesis tool.
The need for new, efficient high-level synthesis techniques is very strong, especially in the SoC era, for the following reasons:
(1) Systems are complex: They can no longer be specified manually at a lower level, such as gate-level and RT-level, where clock cycles need to be detailed. Higher-level abstraction models are required to master the complexity.
(2) Systems are heterogeneous: New specification and design methods are needed to handle the cases where different languages and design style may be used within the same design.
(3) Complex architectures are required to implement future electronic systems: These may include several processors organized around a hierarchical bus structure using a distributed memory structure. A new generation of system design methods allowing the integration of heterogeneous components is required.
(4) Interconnection delay must be considered in high-level synthesis to achieve high performance: The outcome of high-level synthesis defines the internal wires in various resources and interconnection wires among the resources. Therefore, wire generation and optimization should be emphasized in the high-level synthesis.

In order to estimate the interconnection information and to reduce the effect of interconnection wires, several studies combined high-level synthesis with floor planning so that physical information is considered in the process of high-level synthesis [35]. The key problem is to find efficient algorithms to handle the complexity of the combined problem. In general, there are two kinds of approaches:
(1) Constructive approaches: which carry out synthesis and floor planning simultaneously, such as the 3D-algorithm and the GB algorithm;
(2) Iterative approaches: which combine high-level synthesis with floor planning by iterating between scheduling allocation and floor planning in order to achieve the target clock rate [35]. However, the existing constructive algorithms are lack of consideration of the global design information, while the iterative approaches have high complexity and consume too much time.

In this work, our goal is to find an efficient method to provide tradeoff between the two approaches. A key emphasis is to reduce the impact of global interconnects on circuit performance. We shall consider the two approaches simultaneously. We plan to find an efficient flow. In particular, we show how to model the global interconnects during scheduling and allocation, and analyze the impact of interconnect delays on circuit performance. Our high-level synthesizer will accept the C/VHDL behavior description described in Section 4.3.1. We shall develop an internal representation which is able to represent both a hierarchy of CDFGs with timing constraints and the geometric information obtained from floor planning. We shall combine floor planning techniques with high-level synthesis. Based the internal representation, the design task is partitioned to a set of functional-units and then floor planned. The objective function includes not only the total cost of the functional-units, but also the number and the length of interconnection wires. In the phases of scheduling and allocation, the behavioral and physical domains will be unified to generate the placement information for each component introduced in scheduling and resource allocation. The result must meet the timing constraints. It is important that the interconnection delays in the data path components are calculated during the phase. By iterating and refining scheduling, allocation and floor planning, we can gradually converge to a clock rate that satisfies the timing constraints, not only during high-level synthesis, but also after the final layout. Finally, interconnect optimization is needed. It is important to analyze the interconnect wires between the controller and data path components, and to minimize their delays. These interconnects are usually longer and have great impact on the overall performance. Techniques such as buffer insertion, wire sizing and spacing, as well as re-timing over global interconnects can be considered.

5.3.4 Retargetable Compiler Infrastructure and Assembler's Generator

The design of a modern microprocessor needs to consider tradeoffs among software technology, system architecture and micro-electronic implementation. Evaluating the microprocessor's performance during this procedure is one of the primary tasks of the simulation environment support for microprocessor designs. It is important for the compiler (assembler) developer to be able to implement corresponding compilers for target machines with decreased cost and shortened time-to-market, while making full use of the characteristics of the new machines, in order to make more accurate and efficient performance simulation and evaluation. As an important part of compiler infrastructure, a retargetable compiler(assembler) can be easily retargeted to a new machine architecture, including the new instruction set, as well as other new features. The main difficulty in developing the retargetable compiler is to effectively describe machine features such as the pipeline, and to generate a corresponding compiler based on this machine definition efficiently, with good optimization. The NCI (National Compiler Infrastructure) builds a complete and convenient compiler-developing platform to facilitate the development of the compilers. The toolkit includes the CSDL (system description language), ASDL (Abstract Syntax Description language), VPO (very portable optimizer), and so on.

In the JBRC (Jade Bird Retargetable Compiler), we first take the abstract syntax tree as an intermediate representation, which is target-independent. This leads to the well-defined structure of our compiler, whose code generator with encapsulated machine-dependent code and data is clearly separated from the front end. Second, we use a code-generator to produce the instruction selector automatically from a compact specification for a specific target machine. This simplifies the work of retargeting JBRC to a new machine. We use the method of BURG (bottom–up

rewrite system) [36] in our system to finish instruction selection. But there are some insufficiencies currently in the JBRC system: there is little optimization in the backend, and the C language set we supported is not complete.

At present, we are involved in the research of a compiler optimizer, especially in the portable and configurable optimizer oriented to the machine feature, such as pipeline and VLIW and other machine resources. At the same time, we are improving the robustness of the front-end to support the complete ANSI C instruction set, aiming at passing all of C programs benchmarks. We shall redefine the intermediate representation in the next step to separate the front-end more efficiently in order to add some optimization to the front-end. There are two challenges in designing an assembler generator:

- **The first:** is to distinguish different rules which can be used to map assemble language to binary code by all appearances. We should design a specification language to describe the rule.
- **The second:** is to generate an assembler that can generate efficient codes for special microprocessors. The New Jersey Machine-Code Toolkit has made good progress along this line.

Efficiency is another goal in our paper. We use technologies to increase the speed of assembling (such as efficient hash algorithms and build-in linker). JBASM also provides interfaces that the user can use to program special functions for special rules. With retargetable ability, however, JBASM's assembling speed is slower than a custom-made assembler. It is mainly due to the lack of optimizer for specification instructions. This is an area that needs improvement. The main objective in the coming year is to extend the JBASM-SL to describe the OBJ file format, so we can separate assembler and linker. Having a retargetable linker, we can increase the speed of translation and reduce the workload of programming an integrity assembler. On the other hand, we should design how to describe the optimization algorithm taking place in the assembling process. How to define the specification language concisely and completely is the focus.

5.3.5 Synthesis and Optimization for DSP Cores

The design of DSP cores will emphasize code density, power consumption, performance, and compiler friendliness. For related CAD technologies, we shall focus on technologies for low power design. The results of our studies will be used to design the Instruction Set Architecture (ISA) of the DSP processor. For the compiler friendliness, we shall study the instruction set description language for retargetability and DSP library porting.

5.3.5.1 Low-Power Design Technology for DSP Cores

Lower power design can be carried out at architecture design, logic design, and physical design levels. In this paper, we shall study low-power design issues in the context of programmable DSP core at the architecture design level. The following topics will be studied:

(1) Design of Power Management Instruction: One effective way to reduce power consumption in a circuit is to turn off sub-circuits that are inactive during certain machine cycles. To this end, we need to partition a circuit into parts so that different parts can be turned off by the control logic. For a programmable DSP core, we shall study component architectures that

allow dynamic circuit partitioning. For such "partitionable" architectures, we shall study and develop their corresponding power management instructions.

(2) Design of Low Power Controller: In signal and image processing, cyclic execution of a few instructions is often the dominant part of a program, because these applications often involve computations built around convolutions, transforms, and other matrix computations. However, the controller of a general-purpose DSP is designed to decode all instructions all at once. We shall study the concept of "coupled controllers." For a coupled controller, only one of the sub-controllers will be activated most of the time, which can, consequently, lead to substantial savings in power consumption.

(3) Synthesis of High-Performance and Low-Power MAC: MAC (Multiply and Accumulate) is one of the most important operations in a DSP. We shall study the problem of synthesizing high-performance and low-power MAC modules.

(4) Low-Power High-Performance Compiler Optimization Technologies: We shall develop a set of compiler optimization technologies with DSP/CPU features to focus on low-power computing without performance penalty. Those technologies will try to reduce DSP/CPU components' activities, voltage transitions on data/instruction bus, etc.

5.3.5.2 Instruction-Level Simulator Generator

A cycle-correct instruction-level (IL) simulator is the first element of the tool chain for a new CPU design. Any behavior pattern's mismatch between the IL simulator and the real processor may cause unpredictable software bugs. The characteristics of our simulator generator are described below.

(1) RTL/JBASM-SL Based Instruction Level Simulator Generator: The IL simulator generator will understand RTL/JBASM-SL to get hardware profiles from target's RTL information and JBASM-SL descriptions to produce a cycle-correct IL simulator.

(2) Common Information Grapping Interface: The series of generated IL simulators will be promised with compatible profiling interfaces. With this common interface, the collections/analysis tools can be used on CPU/DSP simulators of the next series.

(3) Common Data Modifying Interface: The series of generated IL simulators will be promised with the capability for run-time instruction/memory/register modification. With this capability, run-time optimization with a variety of issues can be loaded with different sequences to evaluate resource benefits.

5.3.5.3 DSP Library Porting

(1) ISDL/Hardware Description Language-Based DSP Library Design: We propose to design a high-performance library based on DSP-understanding compiler derivatives. A compiler with such features can extract information from hardware architecture profiles to generate more efficient machine codes.

(2) Virtual Machine-Based DSP Library Design: We propose to design an environment that hides the details of the underlying real DSP architecture from the designer/programmer. This approach enables developers to write code in an ideal full feature DSP environment. The compiler will fill the missing features between the target architecture and the virtual DSP environment with predefined and efficient subroutines.

5.3.6 Synthesis and Technology Mapping for Embedded FPGAs

Much progress has been made on FPGA synthesis and technology mapping in the past decade. For example, the RASP synthesis system for SRAM-based FPGAs developed at UCLA includes many advanced FPGA synthesis algorithms, such as depth-optimal mapping, duplication-free area-optimal mapping , mapping with resynthesis, simultaneous mapping with optimal retiming, and so on, for generating highly optimized lookup-table (LUT) based netlists. It also includes a set of architecture-specific technology mapping routines to map a generic LUT network to a network of programmable logic blocks (PLBs) of various SRAM-based FPGA architectures, including those with a cluster of LUTs, an array of heterogeneous LUTs, or an array of uniform LUTs with embedded memories (such as Flex 10K devices from Altera). The focus of our paper, however, is on fast incremental synthesis for FPGAs. Including embedded FPGAs in a SoC provides a convenient way to accommodate design changes and support design iterations. As much as we strive for fast design convergence, design iteration is unavoidable due to the ever-increasing design complexity and rapid changing market considerations. Efficient ways to handle minor changes in the design (such as modifications to the netlist or FSM) will be much needed. We will focus on the following two directions:

(1) Efficient incremental synthesis techniques: First, we shall develop a set of efficient synthesis algorithms to handle various incremental changes so that we can satisfy various design constraints while making minimal perturbation to the original synthesis solution. For example, when a new state or a new transition edge is added to an FSM, we need to come up with a new state encoding solution so that the resulting Boolean network is close to the original one, yet meets the area and delay constraints. Existing incremental synthesis techniques are based on local restructuring, such as adding/removing redundancy, reconnecting wires based on internal symmetries, or re-synthesizing nodes and wires using SPFDs. These techniques have been limited to combinational circuits and used in the past on an ad-hoc basis. They need to be studied more systematically, not only in the context of combinational circuits, but also for sequential circuits, with consideration of the rich flexibility provided by programmable logic elements and programmable routing architecture. We need to develop efficient algorithms for partial resynthesis for incremental netlist changes under the performance and area constraints (imposed by higher-level design tools) to achieve the following three levels of objectives for the incremental synthesis:

***Level 1*:** No change in the topology of the mapped programmable logic block (PLB) netlist, and thus no change in the place-and-route solution, or

Level 2: No introduction of new PLBs, but minimal changes in the interconnects, or

Level 3:Minimal changes in the technology-mapping and place–and-route solution.

The main objective is to minimize the effort of incremental place-and-route in the subsequent steps. The system automatically retargets for the next level objective (say Level 2) once it concludes that the current level objective (say Level 1) is not possible to meet. The incremental design changes in the netlist will be decomposed into a sequence of *primitive change operations*, such as adding or deleting a connection, adding or deleting a gate, or changing the function of a gate.

(2) Design for incremental synthesis: If design iteration is unavoidable, it is important to develop a new design methodology with the objective to accommodate incremental synthesis as much as possible. For example, we shall investigate how to come up with a state encoding

scheme for an FSM so that it can accommodate as many incremental changes to the FSM as possible without completely re-encoding and re-synthesizing the FSM. Similarly, given certain area slack in a design, we shall study how to distribute the set of functional blocks into a floor plan solution so that it can accommodate a maximum amount of incremental layout changes without drastically changing the floor plan topology. We need to develop metrics to quantitatively measure the ability to accommodate incremental changes and a new design methodology to optimize such a metric. We shall first develop the theory, general techniques, and metrics to support incremental synthesis and design for incremental synthesis. Then, we shall apply our results to develop an efficient incremental synthesis tool for the target FPGA architecture. In particular, we shall develop an efficient incremental synthesis algorithm for the k/m-macrocell based FPGA architecture.

5.3.7 Synthesis Techniques for IP-Reuse

Due to the rapid advance of fabrication technologies, silicon capacity is doubling every 18 months. This allows companies to build more complex systems on a single chip. However, the ability to develop such complex systems in a reasonable amount of time is diminishing with the rapid increase in complexity. The gap between silicon capacity and design productivity seems to be widening at an ever increasing pace, slowing the growth of the semiconductor industry. In order to solve the productivity gap problem, three approaches have been proposed:

(1) Platforms: In the platform approach, semiconductor vendors provide a universal SoC platform consisting of one or more core processors and many smaller peripheral processors for handling different I/O protocols and encoding/decoding functions.

(2) Reuse: In the reuse approach, it is assumed that SoC designs will be assembled from many different blocks of IP provided by a variety of sources.

(3) Synthesis: In the synthesis approach, the SoC chip will be synthesized from a high-level functional description and tuned for particular applications and fabrication technology.

In our work, we shall focus on the reuse approach. In the reuse approach, it is assumed that SoC designs will be assembled from many different blocks of core/IP (Intellectual Property) provided by a variety of sources. In today's ASIC design flow, reusing RTL components is a common practice. However, reusing simple RTL components may not be adequate in alleviating the design complexity of SoC designs. In order to reduce design complexity and shorten design-cycle time, reusing complex cores becomes a necessity for SoC designs. However, reusing complex cores for a design project is not a trivial task. For this reason, one need to investigate the following topics:

(1) The requirements for reusing a complex IP and how to capture the functional and performance characteristics of IP for reuse. We shall develop a complex IPs/cores capturing method and tool.

(2) How to apply higher-level synthesis techniques (e.g., behavioral level) to facilitate the complex IP/core reuse process, and perform interface generation and SoC integration. We shall develop a synthesis method and tool for automatic IP reuse and system integration.

(3) Develop a C-VHDL-based behavioral specification to silicon design flow. Case studies will be conducted to demonstrate the degree of productivity improvement on the proposed IP-reuse method.

5.3.8 Physical Synthesis for Full-Chip Assembly

In this work, given the limitation on resources, we are not able to include a complete research program on logic-level synthesis and physical design. We plan to use off-the-shelf commercial tools for logic synthesis and physical design of ASIC components. We shall focus our work on physical synthesis for full-chip level assembly, which is a critical step in integrating heterogeneous components and technologies into a single SoC design. But this step is not well understood and there is no existing solution. Full chip level assembly is particularly important for achieving high-performance designs as the global interconnects are bottlenecks in determining system performance in deep submicron technologies. Our work in this area includes floor planning and interconnect planning, power/ground network design, and full-chip level parasitic extraction and modeling.

5.3.8.1 Floor planning and Interconnect Planning

Floor planning and placement of block level is a very important part in the full-chip level assembly. Rapid increase of IC capacity makes it possible to integrate microprocessors, embedded memories, application-specific integrated circuits (ASICs), field-programmable logic arrays (FPGAs), and various analog components into a single IC. The floor planning and block placement tool will place various modules and minimize total chip area and interconnect cost considering timing, noise, power, and specified topology/geometry constraints. Many methods have been developed for floor planning and block placement, which can consider various topology constraints, such as boundary constraints, abutment constraints and fixed block constraints. But there are few studies considering timing, noise and power constraints. We have a new topological representation, corner block list for non-slicing floor plan, which has O(n) complexity. Based on the corner block list and other representation, we have proposed several algorithms in floor planning and placement with boundary constraints, and fixed block constraints with L/T-shape blocks. As the IC technology moves to nanometer dimension and gigahertz frequencies, interconnects play the dominating role in determining the performance, power, reliability, and cost of the system. Therefore, it is necessary to consider interconnect estimation and planning at every design level. Physical-level interconnect planning is to find the best interconnect topology, wire ordering and width, wire spacing, layer assignment, etc., for all global and semi-global interconnects to meet the required performance. UCLA has done an extensive and in-depth study on the design and optimization of high-speed VLSI interconnects in deep submicron designs, and achieved many interesting and original results, including interconnect topology optimization with buffer insertion, optimal wire sizing and spacing, high-speed clock net routing, novel approaches to interconnect performance estimation and design planning, interconnect-centric design flow, and so on [37]. Finally floor planning and interconnect planning for full-chip assembly includes:

(1) Improve corner block list layout representation and speed up floor planning and placement.

(2) Handle multiple constraints, such as boundary, abutment, fixed block, L/T-shape constraints simultaneously.

(3) Consider timing, noise and power constraints, and buffer positions inserted by clock net synthesis.

(4) Consider the effect of parasitic inductance in interconnects to improve and extend research results of design and optimization of high-speed GSI interconnects.
(5) Consider interconnect planning in floor planning and interconnect planning driven floor planning.
(6) Incorporate the recent research results from UCLA on interconnect planning into the floor planning package, including those on interconnect performance estimation (IPEM), wire width planning, buffer block planning, etc.

5.3.8.2 P/G Network Design

With the advance of deep submicron IC technologies, power supply design is becoming an important problem because it affects the performance of circuits considerably. There are two important problems in power/ground network design:
(1) Undesirable wear-out of metal wiring caused by electro-migration, and
(2) Narrowing margins caused by voltage drops.
Power/ground network synthesis finds a topology for a power/ground network, and then minimizes the wiring area of the power/ground network under voltage drop and current density constraints. The basic idea is to transform a constrained nonlinear programming problem into a sequence of linear programs. Like Mitsuhashi's algorithm, voltages and currents are also used as variables. However, all the methods have been tested only on several small test cases. Even the biggest one has only 10,000 nodes. The running time yet is quite long. Many organizations has been working on power/ground network synthesis for many years and proposed several efficient algorithms for the design and optimization of tree-based topology for BBL (building block layout) mode and mesh-based topology for standard cell mode. We plan to work on the improvement of optimization algorithm for mesh-based topology and developing a power/ground synthesis tool. Specific tasks include:
(1) Improve the optimization algorithm based on mathematical programming using conjugate gradient method, circuit sensitivity analysis and equivalent network.
(2) Develop a power/ground synthesis tool, which can handle tree based and mesh based topology of power/ground network.
(3) Consider Delta-I noise by parasitic inductance, multi-pad/multi-tree, movable pads.

5.3.8.3 Parasitic Extraction and Modeling

In deep submicron IC designs, we see the use of very complicated geometry and shape of the interconnect wires, and very high clock frequency (multi-giga hertz). These features lead to the strong proximity effect and skin effect in interconnects. The National Technology Roadmap for Semiconductor shows that in ten years the characteristic dimension will shrink to about 70 nm, clock frequency will achieve 2.5 GH, and the number of interconnect layers will be around nine. At that time, compared with parasitic capacitance, the effect of the parasitic resistance and inductance will affect the performance of GSI circuits in a more significant way. Even now, accurate and efficient extraction of the frequency-dependent resistance and inductance has found many significant applications, such as the clock distribution analysis, non-ideal semiconductor substrate analysis, interconnect analysis in packaging, massively coupled interconnect problems in RF circuits [38], and so on. In the 1990s, many advanced numerical algorithms, such as the MultiPole Accelerated algorithm (MPA), hierarchical computation and parallel MPA

computation, etc., were proposed for capacitance extraction. At the same time, many capacitance extraction tools with good accuracy and reasonable runtime were developed and commercialized by EDA vendors. Compared to capacitance extraction, the study of resistance and inductance extraction was not very active. Currently, there are two kinds of inductance extraction methods:
(1) The volume element method and
(2) The boundary element method.
The well-known extraction tool was based on the volume element method. Compared with the volume element method, the boundary element method has several advantages, such as fewer discrete variables, more accurate model and etc. Hence it has received much attention. In the coming 5 to 10 years, because of requirements in the design of high-performance SoC circuits, in addition to the parasitic capacitance extraction, it will be very important to extract 3-D interconnect inductance and resistance for interconnects with complicated geometry and under 1-4GHz of operating frequency. At the same time, due to higher frequency and more complicated 3-D geometry, it is necessary to find more accurate mathematical-physical models that consider the eddy current in addition to the skin and proximity effects and develop more advanced algorithms with higher computational accuracy and speed. Finally the work on parasitic inductance extraction includes:
(1) Determine mathematical-physical model of inductance extraction considering eddy current and skin effect.
(2) Study the relevant advanced algorithms including fast-solving method for linear equation systems.
(3) Develop a prototype of the inductance extraction tool.

Chapter 6: Emerging Resolutions

This chapter at a glance:

6.1 Reuse and Integration

6.1.1 Why Reuse and Integration?

Over the past ten years, as integrated circuits became increasingly more complex and expensive, the industry began to embrace new design and reuse methodologies that are collectively referred to as system-on-chip (SoC) design. In this section, we focus on the reuse and integration issues encountered in this paradigm shift. The reusable components, called intellectual property (IP) blocks or cores, are typically synthesizable register-transfer level (RTL) designs (often called soft cores) or layout level designs (often called hard cores). The concept of reuse can be carried out at the block, platform, or chip levels, and involves making the IP sufficiently general, configurable, or programmable, for use in a wide range of applications. The IP integration issues include connecting the computational units to the communication medium, which is moving from ad-hoc bus-based approaches toward structured network-on-chip (NoC) architectures. Design-for-test methodologies are also described, along with verification issues that must be addressed when integrating reusable components.

The semiconductor industry has continued to make impressive improvements in the achievable density of Giga-scale integrated (GSI) circuits. In order to keep pace with the levels of integration available, design engineers have developed new methodologies and techniques to manage the increased complexity inherent in these large chips. One such emerging methodology is system-on-chip (SoC) design, wherein predesigned and preverified blocks often called intellectual property (IP) blocks, IP cores, or virtual components are obtained from internal sources, or third parties, and combined on a single chip. These reusable IP cores may include embedded processors, memory blocks, interface blocks, analog blocks, and components that handle application specific processing functions. Corresponding software components are also provided in a reusable form and may include real-time operating systems and kernels, library functions, and device drivers. Large productivity gains can be achieved using this SoC/IP approach. In fact, rather than implementing each of these components separately, the role of the SoC designer is to integrate them onto a chip to implement complex functions in a relatively short amount of time. The integration process involves connecting the IP blocks to the communication network, implementing design-for test (DFT) techniques and using methodologies to verify and validate the overall system-level design. Even larger productivity gains are possible if the system is architected as a platform in such as way that derivative designs can be generated quickly. The purpose of this section is to address the reuse and integration issues in SoC design today. In the past, the concept of SoC simply implied higher and higher levels of integration. That is, it was viewed as migrating a multichip system-on-board (SoB) to a single chip containing digital logic, memory, analog/mixed signal, and RF blocks. The primary drivers for this direction were the reduction of power, smaller form factor, and lower overall cost. It is important to recognize that integrating more and more functionality on a chip has always existed as a trend by virtue of Moore's Law, which predicts that the number of transistors on a chip will double every 18–24 months. The challenge is to increase designer productivity to keep pace with Moore's Law. The increasing separation between the two lines (the compound annual growth rate of transistors on a chip (58%) with the productivity growth of the designer in terms of transistors per month (21%)) is referred to as the productivity gap. The implementation of a chip at the 250-nm node with 50 million transistors provided major challenges; it is

significantly more complex at the 90-nm node, where designs may contain up to 500 million transistors [31]. Therefore, today's notion of SoC is defined in terms of overall productivity gains through reusable design and integration of components.

6.1.2 Reusable IP

The main prerequisite for creating GSI SoC designs is a set of reusable IP blocks that support plug-and-play integration. IP blocks are the highest level building blocks of an SoC. A library of reusable IP blocks with various timing, area, power configurations is the key to SoC success as it allows mix-and-match of different IP blocks so that the SoC integrator can apply the tradeoffs that best suit the needs of the target application. The process of creating a reusable IP block, however, differs from the traditional ASIC design approach. Typically, it may require five times as much work to generate a reusable IP block compared to a single-use block [39]. In the sections to follow, we provide an overview of the issues in design issues for reusable digital IP, analog IP, and programmable IP.

6.1.2.1 Digital IP

Digital IP blocks are the most popular and ubiquitous form of reusable IP in industry today. Many of the tools and design methodologies for creating digital IP were already in place when the concepts of reusability emerged. However, there have been wholesale changes in the design flow to fully enable reusable design. In addition, there are many technical issues that need to be addressed, as the IP developer must anticipate all of the applications in which the IP block may be used. The three stages in the design process are as follows:

- Specification and documentation of the reusable IP;
- Implementation using standardized coding practices;
- Full verification including code coverage and behavioral (or functional) coverage.

The first step includes the generation of suitable documentation for the IP block. The second step includes code design, synthesis, and design for test. Somewhat surprisingly, the second step of implementation is only a small part of the reusable IP design process. In fact, depending on the size and type of the IP, the third step of verification may take up to 50% of the total time. Since the IP may be reused many times in a variety of unanticipated applications, design errors within the block cannot be tolerated. The goal of verification is to achieve 100% code coverage and close to 100% functional coverage to ensure the IP block works correctly. The actual form of a reusable IP core can vary depending on the way in which the IP developer/vendor chooses to provide the core to the system designer. There are three main categories: soft, firm, and hard. These forms are described below and their relationships and tradeoffs are depicted in Figure 6.1.

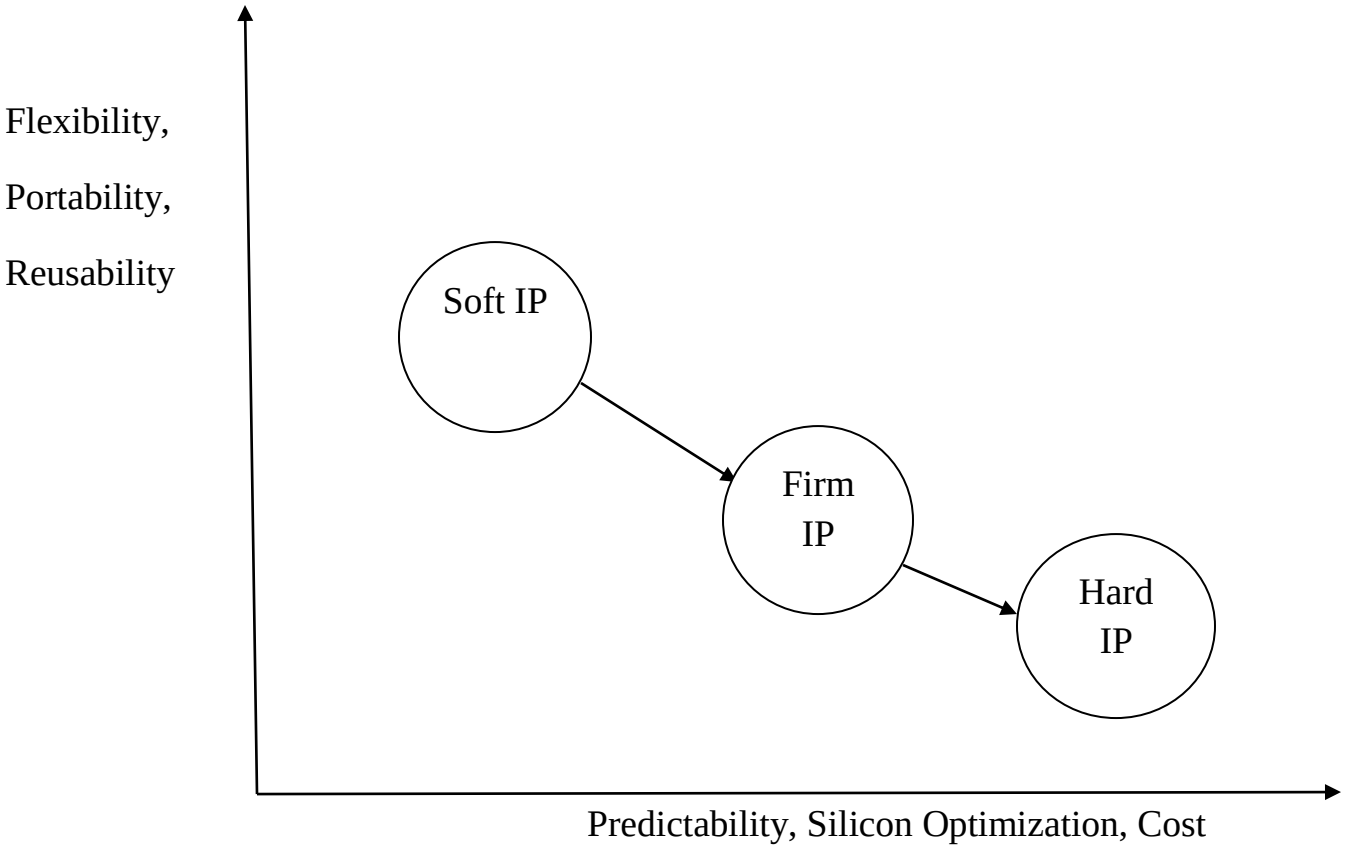

Figure 6.1: Different Types of IP Blocks

• **Soft IP:** Soft IP blocks are specified using RTL or higher level descriptions. They are more suitable for digital cores, since a hardware description language (HDL) is process-independent and can be synthesized to the gate level. This form has the advantage of flexibility, portability, and reusability, with the drawback of not having guaranteed timing or power characteristics, since implementation in different processes and applications produces variations in performance. Sometimes, the HDL is obtained from a third party in an encrypted form which does not allow it to be modified. This makes it harder to adapt it for use in an unanticipated way, but it also prevents the user from introducing any new design errors into the block.
• **Hard IP:** Hard IP blocks have fixed layouts and are highly optimized for a given application in a specific process. They have the advantage of having predictable performance. This, however, comes with additional effort and cost, and lack of portability that may greatly limit the areas of application. This form of IP is usually prequalified, meaning the provider has tested it in silicon. This adds some assurance to its correctness.
• **Firm IP:** Firm IP blocks are provided as parameterized circuit descriptions so that designers can optimize cores for their specific design needs. The flexible parameters allow the designers to make the performance more predictable. Firm IP offers a compromise between soft and hard, being more flexible and portable than hard IP, yet more predictable than soft IP.

Until very recently, most digital IP blocks came in the form of hard IP. For example, ARM and MIPS core vendors would provide behavioral models and black-box layouts of the processors for use during design and verification, and then drop in the hard IP at the foundry facility before fabrication. This afforded the vendor some level of IP protection while allowing the customer to carry out designs using the IP. More recently, soft IP has become the preferred handoff level. Typical soft IP include interface blocks (USB, UART, PCI), encryption blocks (DES, AES), multimedia blocks (JPEG, MPEG 2/4), networking blocks (ATM, Ethernet), and microcontrollers (HC11). The RTL descriptions usually are configurable in that certain parameters are user-definable to tailor the block to the needs of the customer. This allows selective inclusion or exclusion of distinguishing features which can impact the final implementation's performance, cost, and power. In the case of a processor core, parameters such as the bus width, number of registers, cache sizes, and instruction set may vary from customer to customer, so the flexibility of soft IP allows for their modification before synthesis. Commercial tools exist for this purpose in which a configurable processor with specific attributes can be automatically generated [40]. Such flexibility provides lower area, power, and improved performance, since the IP block can be tuned for each application. In addition, some features in a given IP block that are not needed by all customers can be removed. With some effort, the final design generated from a soft IP block can be within 20%– 30% of the power and timing of a hard IP implementation.

6.1.2.2 Analog IP

While design productivity can be improved significantly with the use of digital IP blocks, another bottleneck exists if the designs include analog and mixed-signal components. Digital design has a well-defined, top-down design methodology but analog/mixed-signal (AMS) design has traditionally been an ad-hoc custom design process. When analog and digital blocks coexist on the same substrate, the analog portion of the design can be more time-consuming to develop even though it may represent a smaller percentage of the chip area. In a few years, it is anticipated that more than 70% of all SoC designs will have some form of analog/mixed-signal blocks [41]. This increase is consistent with the expected growth of the wireless industry over the same period. In order to keep pace with rapidly evolving markets, the productivity of AMS design can be improved using a mixed-signal SoC design flow employing AMS IP. One of the main advantages of the use of AMS IP in SoCs is the potential reduction in power, which is especially important in battery-operated applications such as personal digital assistants (PDAs), wireless local area networks (LANs), etc. Typical AMS components include operational amplifiers, analog-to-digital converters (ADCs), digital-to-analog converters (DACs), phase locked loops (PLLs), delay-locked loops (DLLs), serializer/deserializer transceivers, filters, voltage references, radiofrequency (RF) modules, voltage regulators, analog comparators, etc. Many of these blocks are delivered in the form of hard IP and targeted to one application in a specific fabrication technology. Therefore, they cannot be easily migrated to other applications and other technologies by the end user. Compared to digital IP, AMS IP must provide an even greater degree of flexibility in the design parameters and performance characteristics. While the general function of an analog block may be the same in different applications, the design specifications may vary widely between the applications. Furthermore, the performance of AMS IP blocks is significantly influenced by parasitic and interactions with the surrounding environment, often in the form of power supply fluctuations and substrate noise effects.

Currently, because of the complexity of analog/mixed signal design and its sensitivity to the surrounding environment, AMS blocks are most commonly presented in the form of hard IP. However, this form has limited scope of applications. Hard IP will reduce the design cycle significantly when the specifications and fabrication processes are identical, but will not greatly improve the design cycle if it has to be modified in any way or migrated to a new process. This calls for a more flexible definition for the format in which the AMS IPs are provided. Firm IP appears to be the most appropriate format to deliver the AMS IP library components. In this form, the IP captures suitable schematics of the analog blocks with parameters that are adjustable to optimize the design for specific applications. Unlike hard IP, this form allows ease of migration of IP from foundry to foundry, customer to customer, and application to application. The traditional flow for AMS design relies heavily on the expertise and experience of the designer. The design process begins with the performance specification of the component for a target application. Ideally, the AMS block should be described at a high level in the form of soft IP as in the digital case. System-level designers typically use tools such as MATLAB for specification and simulation. In addition, analog/mixed-signal hardware description languages (AMS-HDL) are increasingly used to model these types of circuits. The languages are a relatively new addition to the design process, and the most commonly used are Verilog-AMS and VHDL-AMS. Automatic generation of AMS architectures from AMS-HDL is still in its infancy because of the large number of variables associated with AMS design. However, the current contribution of these AMS-HDLs to system-level design is highly significant. They provide the necessary platform for system level verification, an important part of design quality. Verification of mixed-signal SoCs requires co-simulation of analog and digital behavioral models to reduce simulation costs. Since AMS blocks cannot be easily synthesized from a high-level specification without low-level support, designers must follow a design process such as the firm IP hardening flow illustrated in Figure 6.2. The starting point of the flow is the set of selected library components that comprise the unoptimized schematic view of the design. This library consists of parameterized reusable components and is an essential part of the design flow. The model parameters are set by an optimization tool to achieve the derivative design specifications. After an architecture is chosen, the firm IP is taken through the IP hardening flow for optimization of the circuit parameters to maximize performance and to generate the final GDSII layout of the block. Proper modeling of the interfaces between the different blocks is important in the design process to account for different effects such as loading and coupling. This is needed to achieve correct performance when used in the overall system-level design. When developing firm reusable AMS cores the usual precepts of a good design must be followed. That is, there should be a good formal specification, a good architectural design and a good circuit implementation. In addition, there are a number of steps that must be followed to achieve reusability. A successful IP block should be parameterized, easily verified through reusable test benches, well documented, and have associated views to ease the derivative design process. Specifically, an AMS IP block should have a behavioral/ analytical view (in AMS-HDL), a parameterized schematic view (transistor level), and a layout view (floor plan). In addition, test benches are needed to validate the performance of the circuit under different operating conditions and at various process corners. They are used as the basis for verification of specifications and for exploration of the design space for the system [39]. It is clear that the development of AMS IP must take a different approach compared to digital IP development. The IPs must be able to handle and transfer both design experience and heuristics from the original

design to subsequent design derivatives. In reality, this constitutes the reusable IP in the analog design process.

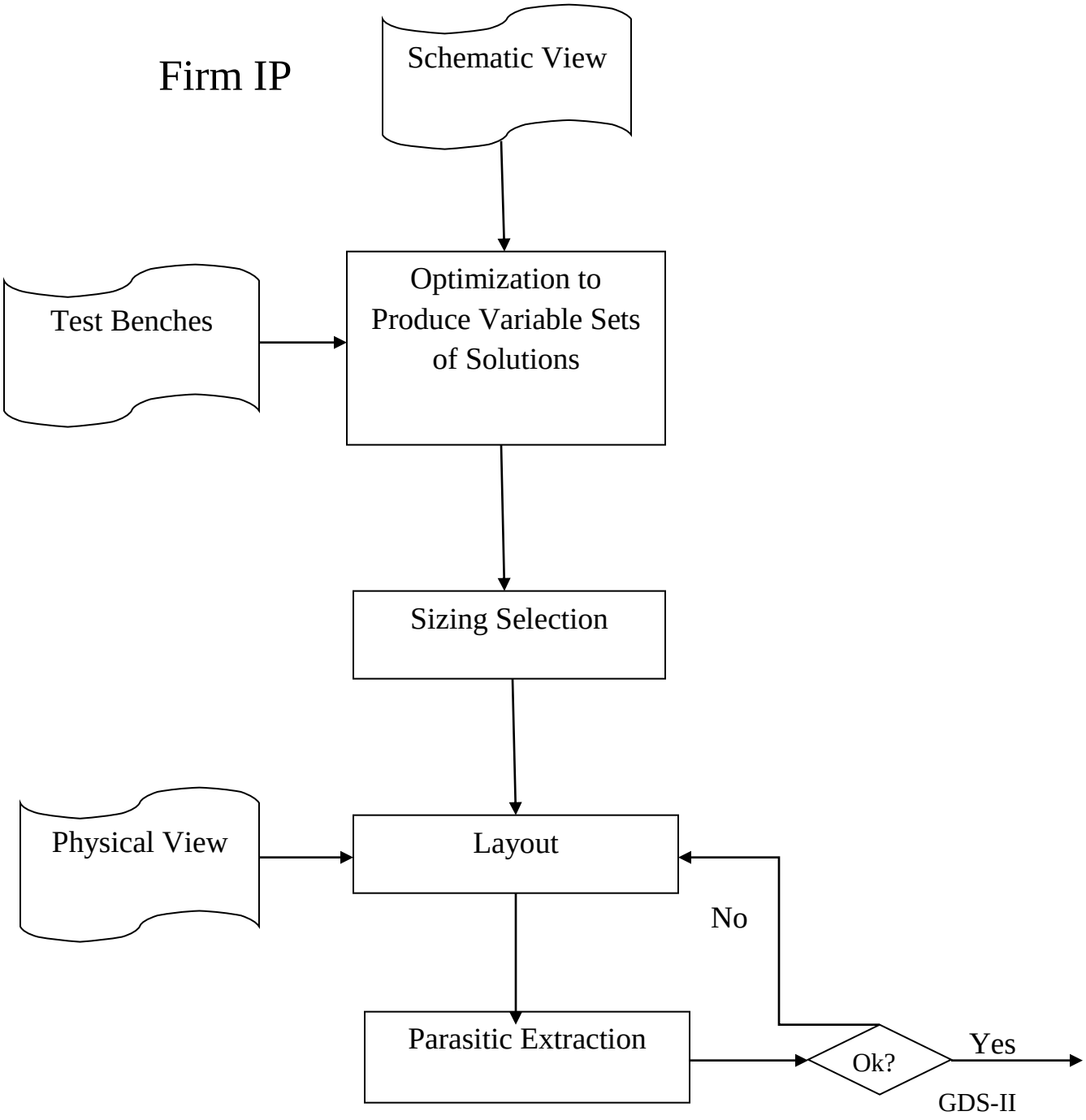

Figure 6.2: Proposed Analog/Mixed-Signal IP Hardening Flow

6.1.2.3 Programmable IP

As SoCs become larger and more complex, the design costs are so high that it becomes important to incorporate programmability within the SoC to allow for reuse at the chip level. This programmability can appear in a number of forms: hardware programmability using programmable logic cores and software programmability using an embedded processor. The key to programmable SoC designs is to provide some form of flexible hardware and/ or software infrastructure, often called the programmable fabric. We distinguish between two types of flexibility:

(1) Prefabrication: Prefabrication flexibility involves the use of a structured ASIC fabric. In this approach, the lower layout layers of the structured ASIC fabric are predetermined and chips are already partially fabricated. The programming is performed by modifying the upper metal and via layout layers just before final fabrication. Since only a few steps are performed to complete fabrication, the turnaround time is relatively short. With this approach, however, the design cannot be changed after the chip is manufactured.

(2) Post fabrication: Post fabrication flexibility allows the behavior of an IP block to be modified after the chip has been fabricated, by either rewriting software/firmware to run on an embedded processor (software configurability) or by reconfiguring embedded programmable hardware, which involves changing the state of configuration bits embedded in the fabric (hardware programmability). The most compelling benefit of post fabrication flexibility is the ability to change the behavior of the IC as requirements change (such as communication standards or customer demands). If the personality [of the design can be changed by rewriting software or reconfiguring the programmable hardware, the creation of a new version of the IC can be avoided; this can typically save several months and significant design costs. The primary disadvantages of post fabrication flexibility (compared to a fixed-function chip) are higher power, lower density (and hence a reduction in the amount of circuitry that can be implemented on a single IC), and reduced speed. However, for many applications, the advantages outweigh the disadvantages.

-> **Hardware Programmability:** Hardware programmability is enabled by the incorporation of one or more programmable logic cores into a SoC [42]. The programmable logic core is a flexible logic fabric that can be customized to implement any digital circuit after fabrication.
Such an embedded core may look very much like the programmable fabric in a stand-alone field-programmable gate array (FPGA). Before fabrication, the designer embeds the fabric (consisting of many uncommitted gates and programmable interconnects between the gates) onto the chip. After the fabrication, the designer can then program these gates and the connections between them

-> **Software Programmability:** Software is the most natural form of programmability for a SoC. Embedded software allows a single SoC to have different personalities and serve different customers or market segments. From a design perspective, as much of the solution should be implemented in software as possible to maximize the flexibility of the design. Complex control functions are better suited to software implementations whereas data operations are better implemented in hardware. If performance is an issue, then hardware must be used for the implementation which can take the form of a programmable logic fabric, which is roughly five

times faster than software, or an ASIC implementation, which is typically 50 times faster than software. Reuse in software is provided through the use of libraries of code and data structures, along with off-the-shelf kernel and real-time operating systems (RTOSs) to improve productivity. Unlike hardware, the software or firmware is never actually completed; it is under continuous development but the code is frozen and released as a version with the intention to update it in the future (and perhaps provide bug fixes when needed). Since SoC programmability is headed in this direction, there will be a growing need for more software/firmware designers in an industry typically dominated by hardware engineers.

6.2 Design and Implementation of Efficient Bus Architecture

6.2.1 Why Design and Implementation of Efficient Bus Architecture?

Today every processing system become very fast and power efficient due to fabrication of multiple system on a single chip .This multiple system need interconnect to communicate with each other through a bus. That‘s why different bus architecture was developed based on different future goal. It also imposes new requirements and challenges. System complexity increases at the same speed. Now a day the system could never be design using the same approaches applied 20 years ago. New architectures are and must be continuously conceived. System-on-chip (SoC) designers today are faced with incredible complexity in the light of billion transistor designs that have already become a reality and ever increasing numbers of components(processors, memories, peripherals, custom hardware) being integrated on a single chip. The onslaught of digital convergence has resulted in requirements for SoC designs which can support more and more functionality (e.g. cell phones with built in MP3 players, digital cameras, AM/FM radios, portable gaming support and PDA functionality) even as the design cycle time keeps shrinking rapidly due to market pressures. Designers today thus have to cope with a large design complexity versus designer productivity gap which continues to widen with each passing year. As billion transistor System-on-chips (SoC) become common place and design complexity continues to increase, designers are faced with the daunting task of meeting escalating design requirements in shrinking time-to-market windows. Since the communication architectures connecting components in these SoC designs significantly impact system performance, it is imperative that designers explore the communication design space efficiently, quickly and early in the design flow. Recent advances in silicon densities now allow for the integration of numerous functions onto a single silicon chip. With this increased density, peripherals formerly attached to the processor at the card level are integrated onto the same die as the processor. As a result, chip designers must now address issues traditionally handled by the system designer. In particular, the on-chip buses used in such system-on-a-chip designs must be sufficiently flexible and robust in order to support a wide variety of embedded system needs. The IBM Blue Logic cores program provides the framework to efficiently realize complex system-on-a chip (SoC)

designs. Typically, a SoC contains numerous functional blocks representing a very large number of logic gates. Designs such as these are best realized through a macro-based approach. Macro based design provides numerous benefits during logic entry and verification, but the ability to reuse intellectual property is often the most significant. From generic serial ports to complex memory controllers and processor cores, each SoC generally requires the use of common macros. Many single chip solutions used in applications today are designed as custom chips, each with its own internal architecture. Logical units within such a chip are often difficult to extract and re-use in different applications. As a result, many times the same function is redesigned from one application to another. Promoting reuse by ensuring macro interconnectivity is accomplished by using common buses for inter macro communications [43].

6.2.2 Types of Bus Architecture

The IBM Core Connect architecture provides three buses for interconnecting cores, library macros, and custom logic:

- Processor Local Bus(PLB)
- On-Chip Peripheral Bus(OPB)
- Device Control Register(DCR) Bus

6.2.2.1 Processor Local Bus (PLB) Architecture

PLB transactions consist of multiphase address and data tenures. Depending on the level of bus activity and capabilities of the PLB slaves, these tenures may be one or more PLB bus cycles in duration. In addition, address pipelining and separate read and write data buses yield increased bus throughput by way of concurrent tenures. Address tenures have three phases: request, transfer and address acknowledge. Allowing PLB devices to move data using long burst transfers can further enhance bus throughput. However, to control the maximum latency in a particular application, master latency timers are required. All masters able to issue burst operations must contain a latency timer that increments at the PLB clock rate and a latency count register. The latency count register is an example of a configuration register that is accessed via the DCR bus. During a burst operation, the latency timer begins counting after an address acknowledges is received from a slave. When the latency timer exceeds the value programmed into the latency count register, the master can either immediately terminate its burst, continue until another master requests the bus or continue until another master requests the bus with a higher priority PLB Cross-Bar Switch In some PLB-based systems multiple masters may cause the aggregate data bandwidth to exceed that which can be satisfied with a single PLB. With such a system it may be possible to place the high data rate masters and their target slaves on separate PLB buses. An example is a multiprocessor system using separate memory controllers. A macro known as the PLB Cross-Bar Switch (CBS) can be utilized to allow communication between masters on one PLB and slaves on the other. As shown in Figure 6.3, the CBS is placed between the PLB arbiters and their slave buses. When a master begins a transaction, the CBS uses the

associated address to select the appropriate slave bus. The CBS supports simultaneous data transfers on both PLB buses along with a prioritization scheme to handle multiple requests to a common slave port. In addition, a high priority request can interrupt a lower priority transaction.

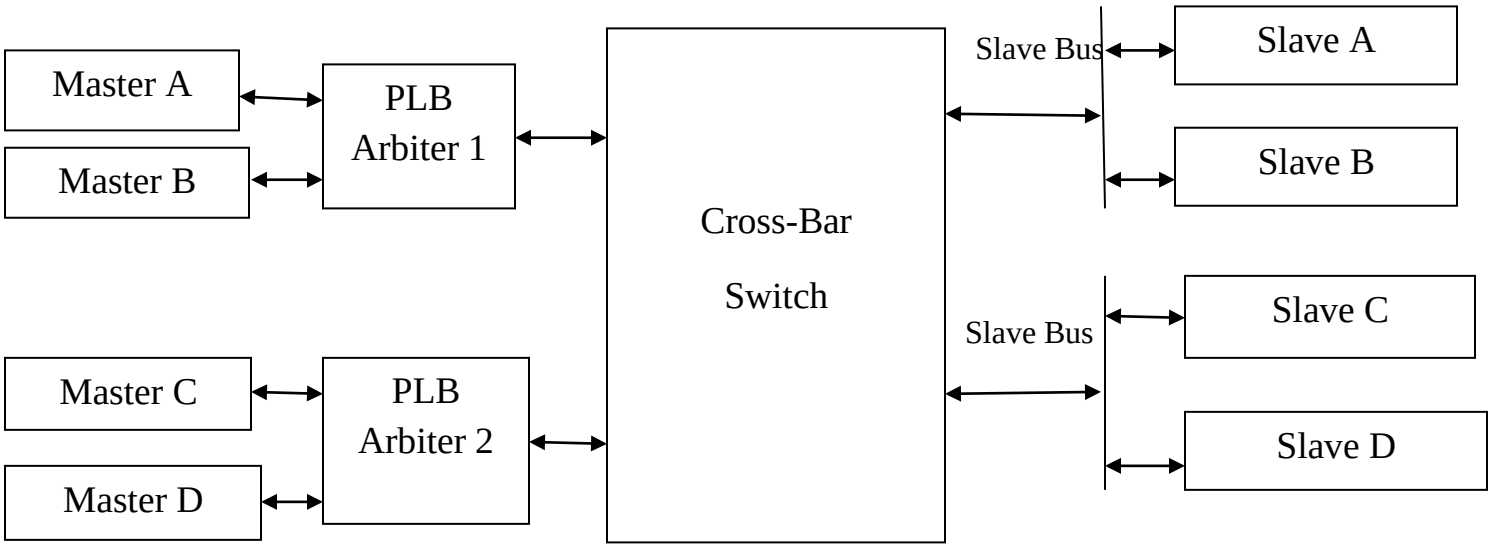

Figure 6.3: PLB Cross-Bar Switch

6.2.2.2 On-Chip Peripheral Bus (OPB) Architecture

The On-Chip Peripheral Bus (OPB) is a secondary bus architecture to alleviate system performance bottlenecks by reducing capacitive loading on the PLB. Peripherals suitable for attachment to the OPB include serial ports, parallel ports, UARTs, GPIO, timers and other low-bandwidth devices. As part of the IBM Blue Logic cores program, all OPB core peripherals directly attach to OPB. This common design point accelerate the design cycle time by allowing system designers to easily integrate complex peripherals into an ASIC. The OPB provides the following features:

- A fully synchronous protocol with separate 32-bit address and data buses
- Dynamic bus sizing to support byte, half-word and word transfers
- Byte and half-word duplication for byte and half-word transfers
- A sequential address protocol
- Support for multiple OPB bus masters
- Bus parking for reduced-latency transfers

6.2.2.3 Device Control Register (DCR) Bus

Lower performance status and configuration registers are typically real and written through the DCR Bus. The DCR provides a maximum throughput of one read or write transfer every two cycles and is a fully synchronous bus typically implemented as a distributed multiplexer. Figure 6.4 illustrates the address and data flow between a processor core master and the DCR slaves. Observe that the relatively slow DCR bus utilizes a ring-type data bus. This provides the required connectivity while minimizing silicon usage.

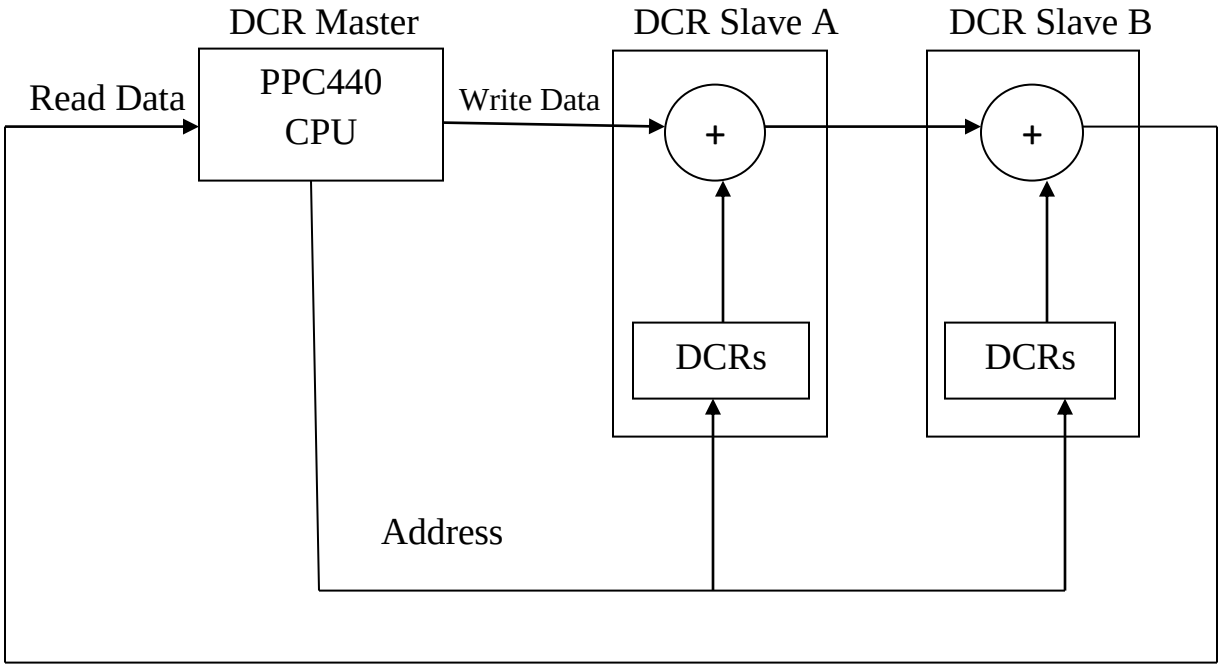

Figure 6.4: DCR Bus

Chapter 7: Verification and Testing

This chapter at a glance:

7. Verification and Testing

Verification and test are increasingly becoming bottlenecks in designing highly complex integrated systems (e.g. GSI SoC). This section discusses future challenges in verification and test, and identifies opportunities for research and innovation [1]. We propose major research efforts in functional verification, test and diagnosis.

7.1 Addressing the Major Challenges in Verification

Due to the technology evolution the capacity per square millimeter and the speed is increasing. These technology improvements allow integrating more functionality and building SoC with high performance, and design becomes very complex. We could easily find SoC with more than 100 IP blocks with over 50% re-use from previous design, and nearly 60-70% of development effort is invested in software. In this complex SoC multicore design is today a fact, where software is distributed across the cores to handle different types of applications. Low power design is unavoidable step where every design team needs to spend a great effort on. Together with the technology evolution, the time to market is becoming even more aggressive than before because all the competing companies need to have a better product and be first in the market. Missing the market window is like "Achilles' Heel". Simulation performance and capacity are improving to keep up with the pace, but without real innovation the inflation point is still to come as the tools or methodologies' limitation will jeopardize the balance.

The key challenges to SoC verification could be summarized as:

- Define the design closure objectives
- Define right test scenarios
- Invest wisely on verification cycles to deal with meaningful verification problem
- Blocks are correctly integrated in the SoC
- Verification must be done in the context of developed or under development Software

Given these challenges, we have identified the following areas that present needs and opportunities for research and innovation [1].

• Hybrid, semi-formal verification

− There is a need for an integrated framework for simulation, emulation and model checking which verifies properties with hybrid techniques.

− We need solutions to automatic test bench generation from behavior code to minimize the manual effort in test bench generation and to improve the quality of simulation-based verification.

− New techniques are needed to improve the efficiency and capacity of formal verification engines for the applications of model checking and symbolic simulation.

• Multi-level/mixed-mode/mixed-technology simulation framework

− We need solutions that support modeling of vastly different functional blocks, predict interference between the blocks and simulate the whole chip accurately within reasonable time.

− Such a framework should also include optimization capability to trade-off among design quality factors such as area, power, signal integrity, and manufacturing yield**.**

For functional verification, research will be conducted in two areas [2]:

(1) Automatic vector generation and assertion checking for HDL, and
(2) Formal verification based on the Extended Finite State Machine model.

The Figure 7.1 highlights the scope, goals and directions of our research in verification.

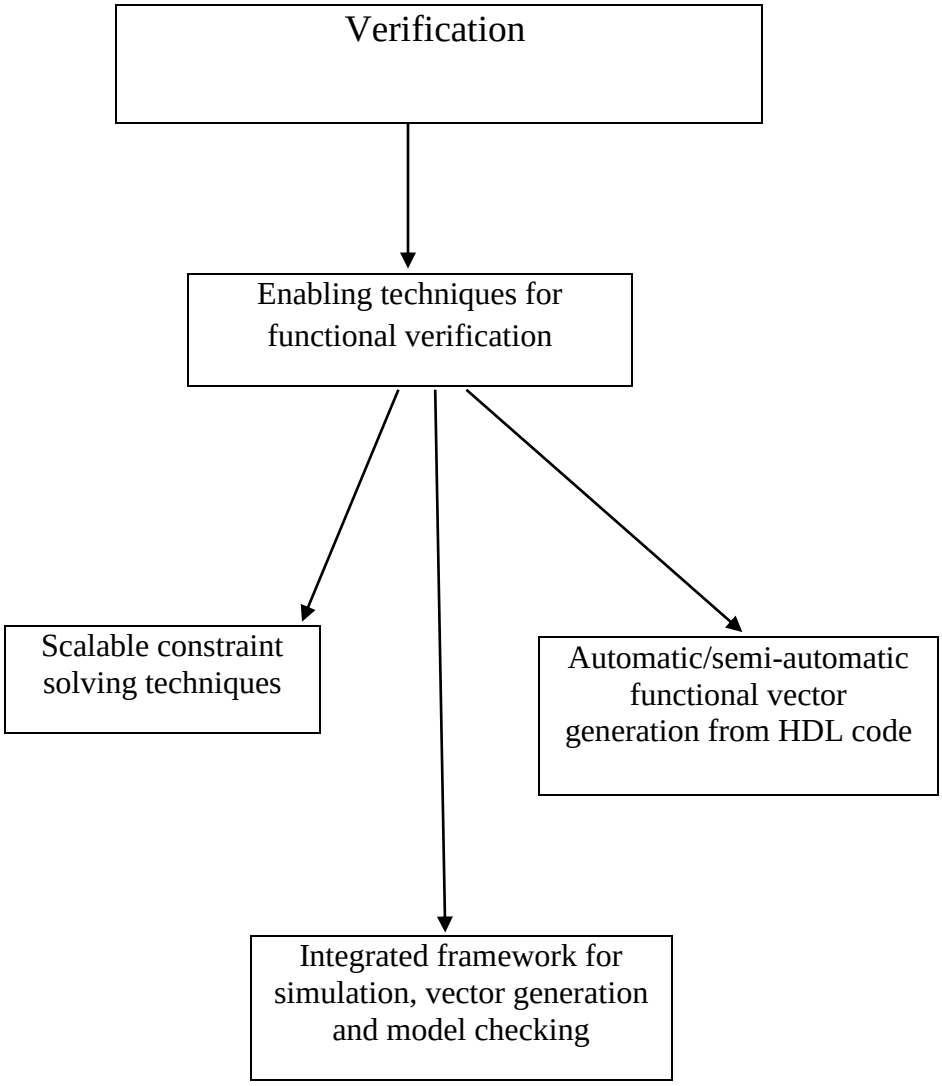

Figure 7.1: Overview of verification

7.1.1 Constraint-Solving for Functional Verification

We propose to develop a new constraint-solving engine for property checking and automatic generation of design verification vectors. The objective is to achieve a more than 10X increase in capacity for model checking and vector generation. The current model checkers are not quite scalable to larger designs. The source of the limitations is primarily in the underlining constraint-solving engine that is the main target of this research. The proposed constraint-solving approach combines structural, word-level automatic test pattern generation (ATPG) and modular arithmetic techniques to solve the constraints imposed by the target property or target coverage metrics. This approach utilizes the strengths of both techniques in solving different kinds of constraints in the circuit and integrates them closely in a unified engine. The word-level ATPG and word-level logic implication techniques primarily solve the constraints imposed by the control logic. They also propagate the logic implications of the assignments made in control logic to the datapath. The new arithmetic constraint solver, based on a modular number system, is then employed to solve the remaining constraints in the datapath. After the initial engine is implemented and thoroughly characterized, we shall then further investigate how to closely integrate BDD and SAT techniques into the engine to further improve the performance of constraint solving. Constraints on the datapath can be divided into two types: linear and nonlinear. We have developed a linear constraint solver based on a modular number system. The linear constraint solver is highly efficient, and finds all solutions and expresses them in a closed form under the modular number system in the complexity of O(n3), where n is the number of input variables. For nonlinear arithmetic constraints, derived from multipliers and shifters, we are exploring some analytical approaches like prime number factoring to heuristically enumerate the possible solutions and substitute them into the arithmetic equations so that the constraints become linear and can be solved by the linear constraint solver. There are several potential advantages of using structural word-level ATPG and modular arithmetic, instead of satisfiability-based (SAT) and linear programming techniques for constraint solving:

(1) We can fully utilize the high-level RTL information and perform word-level implication on both Boolean and arithmetic gates. Several techniques used to translate the implications between Boolean and arithmetic gates have been developed such that conflicting implications can be detected early;

(2) Because the signal values in circuit netlist (RTL netlist) are finitewidth bit-vectors, solving arithmetic constraints in a modular, instead of a general integral number system, will not miss the solutions that come from the modulation and therefore can avoid the false negative effect in generating counter examples ;

(3) The abstract state variables in the Extended Finite State Machine (EFSM) model serve as good candidates of decision points in the branch andbound process of ATPG. In addition, whenever the search encounters a conflict in an abstract state transition or learns that a transition can lead to a hard-to-reach state, the transition in the Extended State Transition Graph (ESTG) is recorded. This recorded information is then used in the subsequent ATPG process to speed up the search;

(4) Compared to the BDD-based symbolic model checking techniques, the proposed method should be much more memory efficient. Both the ATPG-based solver and the arithmetic solver are much less sensitive to the exponential growth of the state space and thus more scalable to larger designs.

7.1.2 Automatic Functional Vector Generation for HDL

Several techniques [12][13] have been proposed to automatically generate test vectors for processor designs, but their usage may be restricted on specific architectures. A technique proposed in can automatically generate functional test vectors for HDL designs using the extended finite state machine model. However, it can handle only the simplest statement coverage. A more general solution is proposed in [14] by using a hybrid satisfiability (HSAT) solver. It has good results on the combinational part of HDL designs but may become too computationally expensive for deep sequential designs because it uses the time-frame expansion to handle the sequential part. In formal techniques, there are many efficient methods to solve the deep sequential problem. Once the *binary decision diagrams* (BDDs) of a finite state machine (FSM) are built, one can perform a *reachability* analysis by the BDD operations. Such analysis, which is the kernel of the symbolic model checking, is still not practical for large designs because of the memory explosion problem of the BDDs. Therefore, suitable partitions to alleviate the peak memory sizes are necessary to handle large designs. Some techniques [15][16] have been proposed to automatically extract FSMs from the HDL designs. With those techniques, one can represent the HDL designs by the interacting FSM (IFSM) model. The desired state transitions in the monolithic FSM are also partitioned into some concurrent transitions of each small FSM in the IFSM model. Most importantly, because the peak memory requirement can be reduced by using the IFSM model and the "divide and conquer" strategy, it can be a practical solution for large designs.

7.1.3 State Enumeration Techniques Based On Extended Finite State Machines

Most existing formal verification algorithms [17][18] suffer from the state explosion problem that is due to the modeling of the entire design using finite state machine models. In this paper, we propose to enhance the capability of property checking from two angles:

(1) We model the design as a set of interacting Extended Finite State Machines (EFSMs) [19]. The EFSM is a more powerful model than the conventional finite state machine and is thus capable of describing complex designs with not only control but also data path components;
(2) We shall develop new state space exploration techniques performed on the EFSM models. The state exploration techniques can be often classified into two types – explicit state enumeration [20], and implicit state enumeration [21].

We shall investigate the extensions of both of these two state enumeration techniques for the EFSM model. The techniques to be developed should be much less prone to the state space explosion problem than other formal verifiers, and thus should be more scalable for larger designs, thereby making formal verification applicable to practical design blocks. The reasons are two-fold. First, the design intentions are nicely captured in the EFSM model, so that the data which registers irrelevant to the property to be checked are automatically abstracted away. Secondly, highly sequential components such as counters are handled differently from the ones for real state register, and thus do not directly contribute to the state space either.

7.2 Challenges and Opportunities in Test

Deep sub-micron technologies and high integration of system-on-chips present challenges to test in a number of areas. The research community must cope with an enormous spectrum of difficult problems ranging from, for instance, high-level test synthesis for core-based design, to noise and power dissipation problems in extremely high performance (in reality analog) pin electronics. The following list highlights some key challenges in test [1].

•**Automatic test equipment is reaching limitations in testing high-performance devices**. On chip clock speed increases dramatically while the tester *overall timing accuracy* (OTA) does not. This trend implies increasing yield loss (estimated to be up to 48% by 2012) as guardbanding to cover tester error results in the loss of more and more good chips. Limited capacity and bandwidth long test application time and excessive cost of ATE also contribute to the increasing overall test cost.
•**Integration of complex, heterogeneous components in SoC posts serious test problems.** Cost-effective methods to test embedded analog, mixed-signal and RF components and devices using multiple technologies, multiple logic families and even multiple voltage levels are needed for such heterogeneous chips. Volume of test data per gate is expected to remain constant, and therefore much more data needs to get across the chip boundary, resulting in bandwidth problems.
•**Deeper submicron technology results in new failure modes.** Process variations are now more likely to cause devices to marginally violate the performance specifications. Testing has to target not only spot defects but also such parametric performance failures. A new class of noise faults caused by excessive noise such as power bus noise, substrate noise, crosstalk-induced delay faults and distributed delay defects needs to be properly modeled for the applications in fault simulation, test generation and design for testability.
•**Paradigm shifts**. Innovative test research is needed to address the challenges outlined above. Without new test methods, testing may become the key showstopper to limit future design and manufacturing technologies. Test technology must be able to support higher-level design and test handoff. Testability must be analyzed/inserted early in the design process of chips and systems. To overcome the ATE limitations, the test systems must become simpler and ICs must be capable of more self-test. Even though ATE will always be required, its role will change in the future. Finally, many of today's deep submicron design validation problems will become tomorrow's test problems, and test research should start addressing them.

Given these challenges, we identified the following areas of needs and opportunities for research and innovation in test in the SoC era [1].

• **Testing and diagnosis for deep sub-micron SoC**
− Defect characterization, fault modeling, testability analysis, vector generation and self-test techniques for emerging DSM defects such as noise- and coupling-induced faults and parametric faults caused by local process variations.
− Diagnostic methodologies/tools for defects in DSM devices for shortening process/design debugging time and for monitoring manufacturing defects.
• **Test solutions for analog/mixed signal/RF circuits**

− BIST solutions for analog and analog-digital interface blocks in mixed-signal designs: For most mixed-signal SoCs, the analog circuitry accounts for only a small fraction of the total silicon area, while a major fraction of the test equipment investment and test time is devoted to the analog parts. Therefore, analog BIST, which could significantly reduce the need for expensive external analog testers, shorten the test time and minimize the measurement noise, offers great promises.
− Modeling, design for testability and BIST for high frequency (to GHZ range) RF circuits.
− On-chip measurement techniques for high-speed serial communication links. Testing devices for applications such as Gigabit Ethernet and Firewire require very expensive test systems capable of generating and receiving differential signals up to several GHz with voltage swings as low as 100 mV. The excessive test time and cost makes existing solutions impossible for volume production.

• **Performance/delay testing**

− Self-testing techniques for delay faults that apply delay tests on chips with greater accuracy than available at the tester.
− Delay test techniques for multi-clock designs (synchronized or not), including on-chip generated clocks.
− Delay test solutions incorporating interconnect, including coupling capacitance, supply line effects and resistive bridges for many levels of metal.
− High speed/frequency testing, paying attention to energy consumption during switching, careful consideration during ATPG of race conditions, multi-cycle delay paths, etc.
− Delay fault models and test strategies for new technologies such as silicon-on-insulator.

• **New current-measurement-based test strategies**

− For deep sub-micron, low voltage, low threshold CMOS circuits: Due to the scaling down of supply voltage and transistor threshold, background IDDQ increases inexorably and the spread of IDDQ distribution is also increasing. IDDQ testing must be adapted to exist in an environment of decreasing signal-to-noise ratio, or be replaced with a better-suited method that maintains its effectiveness in defect screening and reliability prediction.

• **System testing for SoCs**

− Testing strategy/standard for embedded IP blocks.
− Full-chip self-test techniques/methodologies for SoC.
− Built-in self-test, built-in self-diagnosis and built-in self-repair techniques for deep sub-micron, embedded memories.
− Test scheduling and power/thermal management for SoC: power consumption during test must be carefully considered so that testing does not put the system into modes that consume excessive power/current or cause excessive noise resulting in soft errors.
− SoC test synthesis tool development.

The Figure 7.2 highlights the scope, goals and directions of our research in testing.

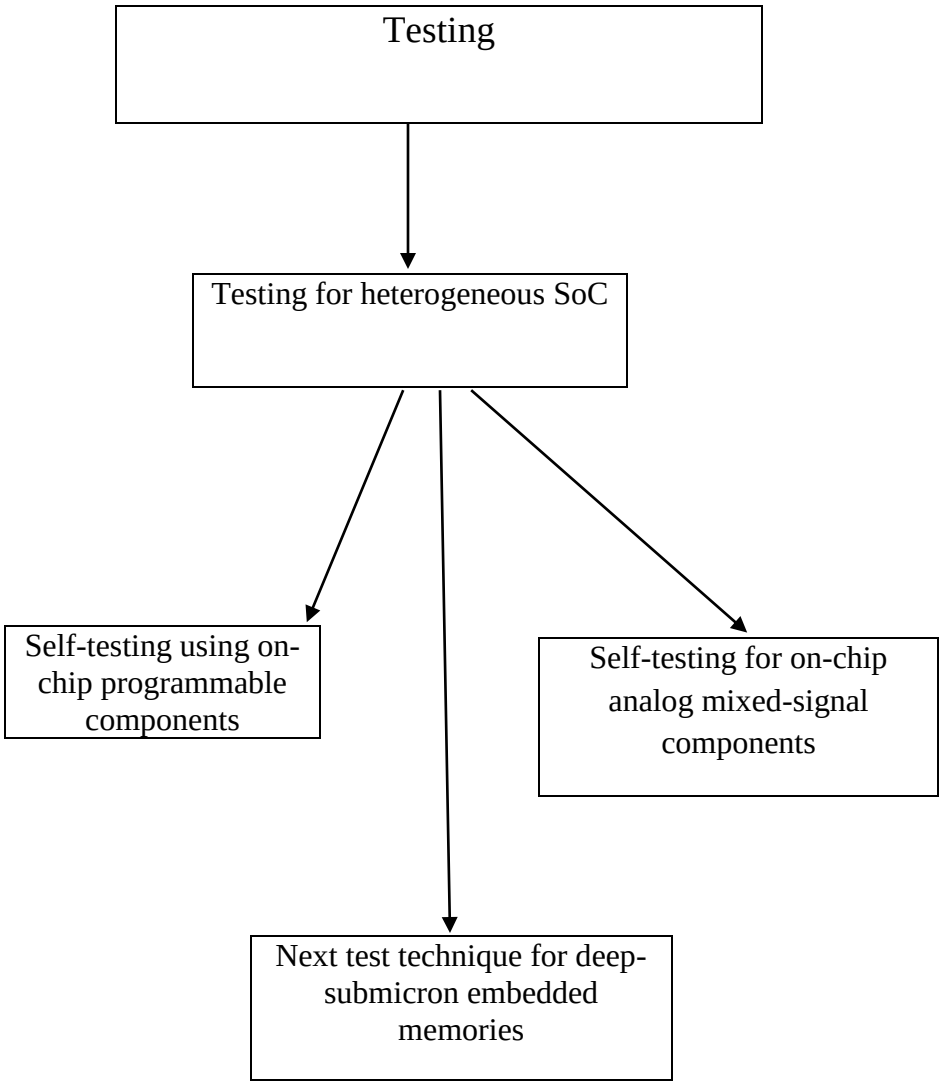

Figure 7.2: Overview of Testing

7.2.1 Self-Test and Self-Diagnosis Using On-Chip Programmable Resources

The strategy of using on-chip programmable cores for self-testing actually views testing as an application of the programmable components in the SoC and thus , minimizes the addition of dedicated test circuitry for DFT or self-test. Our goal is to provide an automated test program synthesis methodology for these applications as well as to understand the capability, capacity and limitations of these test applications. Existing logic BIST techniques belong to the class of structural BIST. Structural BIST such as the scan-based BIST technique [22][23], offers good test quality, but requires addition of dedicated test circuitry (such as full-scan, LFSRs for pattern generation, MISRs for data analysis and test controllers). Therefore, it incurs non-trivial area, performance and design time overhead. Moreover, structural BIST applies non-functional, high-

switching random patterns and thus, causes much higher power consumption than normal system operations. Self-testing the devices using an instruction set of on-chip processors has the potential to alleviate these problems. We refer this self-test strategy as a functional self-test or embedded software tester solution. To apply at-speed tests to detect timing-related faults, existing structural BIST needs to resolve various complex timing issues related to multiple clock domains, multiple frequencies and test clock skews which are unique in the test mode. In contrast, self-testing the devices using instruction sequences allows more natural application of at-speed tests. Delay tests are applied by executing instruction sequences in a similar fashion as normal system operations. Our paper thrust is to develop methods for automatic synthesis of test programs which, when executed, result in high delay fault coverage for target delay defects. Pure functional self-test may or may not give a desired level of test quality. We propose to conduct research to characterize the capability and limitations of functional self-test and gain more insights into how functional self-test can complement structural BIST. We then further conduct new research to study hybrid solutions that combine the strengths of functional and structural self-test. We shall further extend these functional self-test techniques to self-diagnosis of processor components. Self-diagnosis programs using the embedded processor's instruction set will be generated on-chip to identify the location of an error in the embedded processor, and programmable delivery mechanisms will be developed to apply the diagnosis tests at the operational speed of the embedded processor.

7.2.2 Analog and Mixed-Signal Self-Test

We further propose to develop efficient self-test techniques that target the analog/mixed-signal components in mixed signal system-on-chip (SOC) designs. Since digital programmable resources such as DSP or processor cores are commonly available in modern mixed-signal SoC designs, our strategy of mixed-signal self-testing is to utilize such on-chip programmable resources while limiting the use of analog/mixed-signal self-test circuitry. This approach not only enhances the flexibility, but also reduces the amount of dedicated test circuitry for self-testing. Among the various analog/mixed-signal Built-In Self-Test (BIST) schemes proposed recently, we are particularly interested in the DSP-based BIST approach. In the DSP-based BIST approach, on-chip analog-to-digital (AD) and digital-to-analog (DA) converters are used for analog stimulus generation and response digitization, and DSP processors are used for DSP operations needed to prepare the digitized test stimulus and perform response analysis. This approach is attractive because:

(1) The needed DSP resources can usually be found in modern mixed-signal SoC design; and

(2) The setup is flexible in that different tests (DC, AC, and dynamic tests) can be applied to the circuit under test (CUT) by modifying the software routines without hardware alternation.

However, it is not uncommon that a mixed-signal SoC has only an on-chip AD or DA converter, but not both, which limits its application. We propose to develop efficient BIST techniques for SoC devices that have on-chip DSP resources but lack an on-chip AD or DA converter. To fully utilize the digital resources, the devised techniques should have the following features:

−Signal processing for the test execution should be performed by the on-chip DSP resources whenever possible.

−The use of analog/mixed-signal BIST circuitry should be minimized.

− The analog BIST circuitry must have robust performance against process variations, i.e., the analog/mixed signal BIST circuitry won't fail at the presence of process variations within a reasonable range.
−One must be able to characterize and test the analog/mixed-signal BIST circuitry relatively easily with on-chip resources and minimum aid from external ATE.

7.2.3 Testing of Embedded Memories in SoC

We propose to investigate test and diagnosis techniques for deep submicron, embedded memories. The topics to be explored include fault modeling for new failure modes, vector generation, built-in self-test and built-in self-diagnosis.

• **New Fault Models for Memories**: In DSM memory cores, we need to consider delay faults and interconnect coupling and noise issues as in logic cores. In addition, reliability failures will be harder to deal with, especially for on-chip DRAM, flash-EPROM, and flash-EEPROM. New fault models such as word-line (gate) disturbance faults, bit-line (drain/source) disturbance faults, read disturbance faults, cell endurance faults, and data retention faults need to be addressed.
• **Vector Generation for Memories**: The March-based test algorithms are considered the most cost-effective for RAM stuck-at, transition, coupling, stuck-open, and address-decoder faults. Traditionally, the development of test algorithms aims at covering as many fault models as possible [24]. However, with the growing complexity of memory chips, test time reduction is more and more a concern, even for linear-time algorithms. Efficient test algorithms for the ever-innovating memory chips will need to be discovered in a much shorter time than before. Test algorithm generation and verification for RAM is conventionally done using mathematical approaches [25]. Though elegant, these mathematical approaches do not provide fault coverage figures during the generation and verification process as well as in the data background selection steps. We shall investigate a different approach: testing algorithm generation by simulation (TAGS) [26], which is based on fast memory fault simulation. Given a set of faults, TAGS can generate a series of March tests with different March elements and lengths. It can be used to optimize the test algorithms with respect to test time or fault coverage. We shall also investigate the possibility of using TAGS to generate test algorithms for disturbance, endurance, and retention faults.
• **Memory BIST and Diagnosis**: For memory cores, built-in self-test (BIST) is a relatively mature approach to attacking their test issues [27]. In addition to testing the embedded memories using March-based algorithms, diagnosis of the fault sites and subsequent repair by redundant word-and/or bit-lines to increase the yield is necessary for large cores. Therefore, built-in self-diagnosis (BISD), built-in redundancy analysis (BIRA), and even build-in self-repair (BISR) methodologies are becoming inevitable, so far as overall test cost is concerned. Diagnosis may also be done for failure/fault analysis. In this case, however, in addition to error location by BIST, fault types should also be identified and classified, followed by required analysis to come up with fault locations. We shall develop BISD (including BIST) methodology and design technology for embedded RAM (SRAM and DRAM).

Chapter 8: Analytical Result and Discussion

This chapter at a glance:

8.1 Analytical Result and Discussion

8.2 Summation

8.1 Analytical Result and Discussion

The success of system-on-a-chip (SoC) hinges upon a well concerted integrated approach from multiple disciplines, such as device, design, and application. From the device perspective, rapidly improving GSI technology allows the integration of billions of transistors on a single chip, thus permitting a wide range of functions to be combined on one chip. From the application perspective, numerous killer applications have been identified, which can make full use of the aforementioned functionalities provided by a single chip. From the design perspective, however, with greater device integration, system designs become more complex and are increasingly challenging to design. Moving forward, novel approaches will be needed to meet these challenges. We explores several new design strategies, which represent the current design trends to deal with the emerging issues. For example, recognizing the stringent requirements on power consumption, memory bandwidth/latency, and transistor variability, novel power/thermal management, multi-processor SoC (MPSoC), reconfigurable logic, and design for verification and testing have now been incorporated into modern system design. In addition, we also provide some plausible solutions. For example, IP deployment and integration, on-chip interconnects, and design and implementation of efficient bus architecture.

We get the overall view of the challenges in physical design that might face Giga-scale integration system-on-a chip design. We have also shown the design challenges for on-chip communication such as: technology issues, performance issues, etc. In future the couplings and interactions among system components will increase as we put more of the system on a silicon die. Therefore the system designers will face challenges in several areas and we provide here the future system design challenges such as: scalable or reusable architecture, IP integration, reliability and so on. Now to overcome these challenges with our today's technology, we need to perform synthesis. Synthesis works as a bridge between the challenges and how they can overcome. Therefore we have shown the overall synthesis mechanism. We have shown the research areas related to synthesis and the synthesis environment. We also provide major research efforts in functional verification, test and diagnosis. We have shown the challenges and opportunities in functional verification and challenges and opportunities in test. Finally we provide two emerging resolutions by which we can overcome the maximum challenges that face today's GSI SoC design.

The overall analysis, from basic driving force to emerging issues, from modern design trends to future challenges, from synthesis to emerging resolutions all are served by the Figure 8.1.

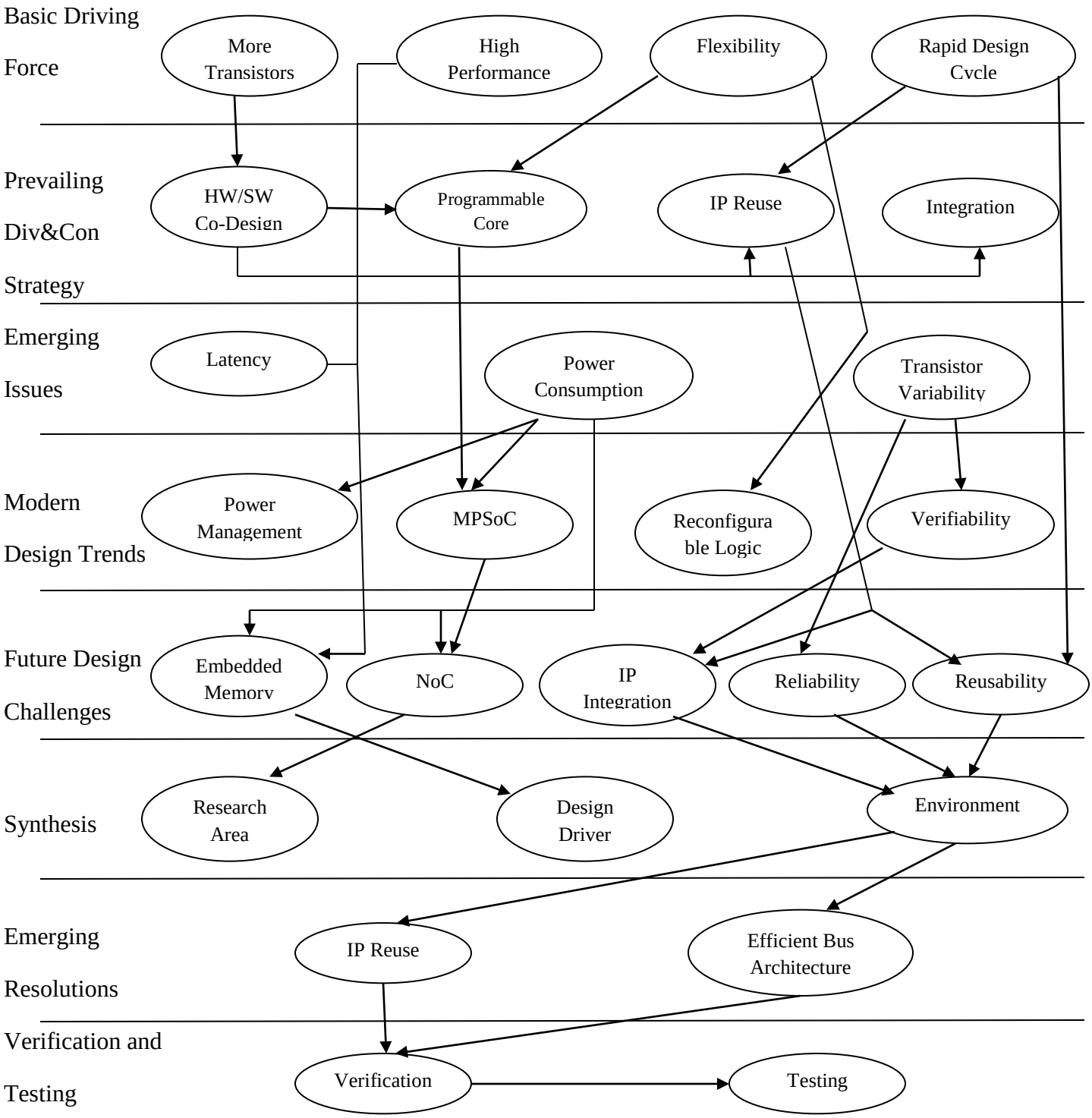

Figure 8.1: Overall topics of GSI SoC design.

Figure 8.1 serves as an outline of this work. The links between boxes reflects the cause-effect relationship. For example, the relationship between the cause-nodes (programmable core, and power consumption) and the effect node (MPSoC) is highlighted here.

1. The top level shows the basic driving forces behind SoC designs, including (1) integrating more transistors within (2) a short period of time to provide (3) high performance and (4) flexibility.
2. The second level shows a divide-and-conquer strategy which was adopted to create flexible SoC products with short design cycles. To elaborate on this strategy, several typical approaches, including hardware-software partitioning/co-design, programmable core, IP design and reuse, and vertical integration can be used.
3. The third level displays emerging issues, including power consumption, memory bandwidth/latency, and transistor variability, which have a major impact on system designs.
4. Recognizing these emerging issues, the fourth level shows how modern system design has incorporated novel power/thermal management, multi-processor SoC, reconfigurable logic, and design for verification and testing.
5. As depicted in the fifth level, we anticipate further innovations on scalable, reusable, and reliable system architectures, IP deployment and integration, on-chip interconnects, and memory hierarchies in the near future.
6. The sixth level is Synthesis which works as a bridge between the challenges and how they can overcome. The study of synthesis include: research area, design driver and synthesis environment.
7. The seventh level provide two emerging resolution which provide high performance and other required criteria for good SoC design. The two resolutions are: (1) IP Reuse; (2) Efficient Bus Architecture.
8. The last level displays the verification and testing which provide the challenges, opportunities and functionality of verification and testing of a design.

8.2 Summation

So after analytical study of the above section and full description of the previous sections, one can easily understand what is GSI SoC, what is its challenges and lackings and what is its solution. By following the path we described in Figure 8.1, one can easily perform an efficient design of GSI SoC.

Chapter 9: Conclusion and Future Work

This chapter at a glance:

9.1 Conclusion

9.2 Future Work

9.1 Conclusion

The exponential scaling of the integrated circuit technology results in the integration of hundreds of millions of transistors on a single chip, which provides the possibility of integration to a highly complex, heterogeneous electronic system on a single chip. To achieve such Giga-scale integration SoC, however, requires a great deal of innovation in the design and test technologies, in terms of both steady incremental extension of the existing design and test capabilities and the revolutionary development of new design and test paradigm and methodologies. Significant investment is needed in these areas in order to extend the Moore's Law into the next two decades.

The technology revolution has had a profound impact on our daily life. For example, personal communication will never be the same. About 80 years ago, Bell Laboratories demonstrated the first mobile phone prototype which had to be mounted on a car. About 30 years ago, the first hand-held mobile phone became commercially available. Today, the standard cellular phone is compact, user friendly, and packed with functionality. Moreover, many optional features are available, such as personal digital assistants and global positioning systems. In fact, more and more new applications are ready to harness the technology revolution. Prominent examples include third-generation wireless communication and a wide range of popular lifestyle consumer electronics (such as, digital cameras, video camcorders, digital televisions, and set-top boxes). According to In-Stat/MDR, the market for smart appliances in digital home experienced a 70% compound annual growth rate from 2001 to 2018. Moving forward, Gartner Market Report predicted that $500 millions market for SoC in 2005 will grow over 80% by 2018. The annual growth rate is about 2x faster than general-purpose microprocessors. Such a change is largely due to the advances in device technology, which enable us to put billions of transistors on a chip for almost unlimited processing capability.

9.2 Future Work

There are different opinions on the future of microprocessor advancement, but the use of terms like Giga-scale integration indicates there is still room for advancement in putting even more logical design into smaller and smaller chips such as Tera-Scale Integration (TSI) where the number of logic gate would be 10^12 or more. The growing complexity of embedded multi-processor architectures for digital media processing will soon require highly scalable communication infrastructures. Packet switched Networks- on-Chip (NoC) have been proposed to support the trend for Systems-on-Chip integration. SoC is evolving along with other technologies such as silicon-on-insulator (SoI), which can provide increased clock speeds while reducing the power consumed by a microchip.

References:

[1] Jason Cong "Challenges and Opportunities in Giga-scale Integration For System-On-A-Chip", NSF/NSC International Workshop, August 24-26,1999.

[2] Proposal to National Science Foundation for Establishing an International Research Center on " Giga-Scale System-On-A-Chip Design", Version 7, 8/11/00.

[3] Bertozzi, D., & Benini, L. (2004). Xpipes: A network-on-chip architecture for gigascale systems-on-chip. *IEEE circuits and systems magazine*, *4*(2), 18-31.

[4] Benini, L., & De Micheli, G. (2002). Networks on chips: A new SoC paradigm. *Computer-IEEE Computer Society-*, *35*(EPFL-ARTICLE-165542), 70-78.

[5] Goossens, K., Dielissen, J., van Meerbergen, J., Poplavko, P., Rădulescu, A., Rijpkema, E. & Wielage, P. (2003). Guaranteeing the quality of services in networks on chip. In *Networks on chip* (pp. 61-82). Springer, Boston, MA.

[6] Top-down SoC Design Methodology, Design and reuse, 2018, https://www.design-reuse.com/articles/4952/top-down-soc-design-methodology.html

[7] Gigascale Integration, Techopedia, 2018, https://www.techopedia.com/definition/7410/gigascale-integration-gsi

[8] Wikipedia, Wikimedia Foundation, System on a Chip, 2018, https://en.wikipedia.org/wiki/System_on_a_chip

[9] Hardware Terms : SoC Definition, TechTerms, 2018, https://techterms.com/definition/soc

[10] ITRS 1999. Available: http://public.itrs.net/files/1999_SIA_Roadmap/

[11] A. Jantsch and H. Tenhunen, "Will networks on chip close the productivity gap?," in *Networks on Chip*, A. Jantsch and Hannu Tenhunen, eds. Dordrecht: Kluwer, 2003, pp. 3–18.

[12][JoLi 99] Jing-Yang Jou and Chien-Nan Jimmy Liu, "Coverage Analysis Techniques for HDL Design Validation," the 6th Asia Pacific Conference on cHip Design Languages (APCHDL'99), October 1999. (Invited Paper)

[13] [ChIy95] A. Chandra, V. Iyengar, D. Jameson, R. Jawalekar, I. Nair, B. Rosen, M. Mullen, J. Yoon, R. Armoni, D. Geist, and Y. Wolfsthal, "AVPGEN – A Test Generator for Architecture Verification," IEEE Trans. on VLSI, vol. 3, no. 2, pp. 188-200, June 1995.

[14] [ChKr93] Kwang-Ting Cheng and A. S. Krishnakumar, "Automatic Functional Test Generation Using the Extend Finite State Machine Model," 30th DAC, Jun. 1993.

[15] [CaCa97] Gianpiero Cabodi and Paolo Camurati, "Symbolic FSM Traversals Based on the Transition Relation," IEEE Trans. on Computer-Aided Design, vol. 16, no. 5, pp. 448-457, May 1997.

[16] [LiJo98] Chien-Nan Liu and Jing-Yang Jou, "A FSM Extractor for HDL Description at RTL Level," The Fifth Asia-Pacific Conference on Hardware Description Languages (APCHDL), Seoul, Korea, July 1998.

[17] [Br86] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Trans. On Computers, Vol. C-35, pp. 677-691, 1986.

[18] [BuCl94] J. Burch, M. Clarke, E. Long, K. L. McMillan, D. L. Dill, "Symbolic Model Checking for Sequential Circuit Verification," IEEE Trans. On Computer-Aided Design, Vol. 13, No. 4, pp. 401-424, April 1994.

[19] [ChKr93] K.-T. Cheng and A. S. Krishnakumar, "Automatic Functional Test Generation Using The Extended Finite State Machine," Proc. Of Int'l Design Automation Conf., pp. 86-91, 1993.

[20] [Ku96] R. P. Kurshan, "Formal Verification of Coordinated Processes," IEEE Spectrum, pp. 61-67, June 1996.

[21] [Mc93] K. L. McMillan, "Symbolic Model Checking," Kluwer Academic Publishers, 1993.

[22] [LiZo95] C.-J. Lin, Y. Zorian, and S. Bhawmik. Integration of Partial Scan and Built-In Self-Test. Journal of Electronic Testing: Theory and Applications, 7(1-2):125-137, August 1995.

[23] [ChLi95] K.-T. Cheng and C.-J. Lin. Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST. Jason Cong Page 34 4/23/01 Proceedings of International Test Conference, pages 506-514, October 1995.

[24] [RiRa95] M. Riedel and J. Rajski, "Fault coverage analysis of RAM test algorithms," in *Proc. IEEE VLSI Test Symp. (VTS*), 1995, pp. 227–234.

[25] [ZaUp98] K. Zarrineh, S. J. Upadhyaya, and S. Chakravarty, "A new framework for generating optimal march tests for memory arrays," in *Proc. Int. Test Conf. (ITC*), 1998, pp. 73–82.

[26] [WuHu00] C.-F. Wu, C.-T. Huang, K.-L. Cheng, and C.-W. Wu, "Simulation-based test algorithm generation for random access memories," in *Proc. IEEE VLSI Test Symp. (VTS*), (Montreal), Apr. 2000 (to appear).

[27] [HuHu99] C.-T. Huang, J.-R. Huang, C.-F. Wu, C.-W. Wu, and T.-Y. Chang, "A programmable BIST core for embedded DRAM," *IEEE Design & Test of Computer*s, vol. 16, pp. 59–70, Jan.-Mar. 1999.

[28] Moore's Law, Investopedia, 2018, https://www.investopedia.com/terms/m/mooreslaw.asp

[29] System-on-a-chip, IoT Agenda, 2018, https://internetofthingsagenda.techtarget.com/definition/system-on-a-chip-SoC

[30] Yen-Kuang Chen, Intel Corporation, Santa Clara, CA 95052 and S. Y. Kung, Princeton University, Princeton, NJ 08544 " Trend and Challenges on System-on-a-Chip Designs" Journal of VLSI Signal Processing Systems, DOI: 10.1007/s11265-007-0129-7

[31] Resve Saleh, Fellow IEEE, SteveWilton, Senior Member IEEE, Shahriar Mirabbasi, Member IEEE, AlanHu, Member IEEE, Mark Greenstreet, Member IEEE,
Guy Lemieux, Member IEEE, Partha Pratim Pande, Member IEEE, Cristian Grecu, Student Member IEEE, and Andre Ivanov, Fellow IEEE "System-on-Chip: Reuse and Integration" IEEE, Vol. 94, No. 6, June 2006

[32] [CoHu00] Cong, J., H. Huang and X. Yuan, "Technology Mapping for k/m macrocell Based FPGAs," *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, February 2000, pp. 51-59.

[33] [CoPe96] Cong, J., J. Peck, and Y. Ding, "RASP: A General Logic Synthesis System for SRAM-based FPGAs," Proc. ACM/SIGDA Int'l Symp. on Field Programmable Gate-Arrays, Monterey, CA., Feb. 1996, pp. 37-143.

[34] [DaMa97] J.-M. Daveau, G. Fernandes Marchioro, and A.A. Jerraya "VHDL generation from sdl specification" In Carlos D. Kloos and Eduard Cerny, editors, Proceedings of CHDL, 182-201. IFIP, Apr. 1997.

[35] [MoTa96] Moshnyaga, Vasily G.; Tamaru, Keikichi. Floorplan based methodology for data-path synthesis of sub-micron ASICs; IEICE Transactions on Information and Systems v E79-D n 10 Oct 1996 1389-1395.

[36] [CWF92] Christopher W. Fraser, R. Henry and T. Proebsting. BURG-Fast optimal instruction selection and tree parsing; SIGPLAN Notices 27(4):68-76, 4/92.

[37] [CoHe97b] Cong, J., L. He, C. K. Koh, and Z. Pan, "Global Interconnect Sizing and Spacing with Consideration of Coupling Capacitance," Proc. IEEE Int'l Conf. on Computer-Aided Design, San Jose, CA., Nov. 1997, pp. 628-633.

[38] [KaMa99] M. Kamon, N. Marques, Y. Massoud, L. Silveira, J White, "Interconnect Analysis: From 3-D Structures to Circuit Models," DAC'99, pp.910-914, 1999

[39] M. Keating and P. Bricaud, "Reuse Methodology Manual: For System-on-a-Chip Designs", 3rd ed. Boston, MA: Kluwer, 2002.

[40] Xtensa Architecture and Performance, White Paper, Tensilica Inc., Sep. 2002.

[41] International Techbology Roadmap for Semiconductors. [Online]. Available:
http://www.public.itrs.net

[42] S. Wilton and R. Saleh, "Programmable logic IP cores in SoC design: opportunities and challenges", in Proc. IEEE Custom Integrated Circuits Conf., 2001, pp. 63–66.

[43] Cordan, B. (1999). An efficient bus architecture for system-on-chip design. In *Custom Integrated Circuits, 1999. Proceedings of the IEEE 1999* (pp. 623-626). IEEE.