CHAPTER 1

Introduction

1.1 INTRODUCTION

The ASIC (Application Specific Integrated Circuit) and SOC (System on a Chip) abbreviations are used every day in the integrated circuit design industry. However, there are still a lot of ambiguities when differentiating SOCs from traditional ASICs. Some designers define SOCs as complex integrated circuits with more than one on-chip processor. Many use the term when describing ICs that have more than 10 million gates plus on-chip processors. Still others define it as ICs that contain soft and hard functional blocks as well as digital and analog components. Let's give our own definition here.

An SOC is a system on an IC that integrates software and hardware Intellectual Property (IP) using more than one design methodology for the purpose of defining the functionality and behavior of the proposed system. In most cases, the designed system is application specific. Typical applications can be found in the consumer, networking, communications, and other segments of the electronics industry. Voice over Internet Protocol (VoIP) is a good example of an emerging market where SOCs are widely designed. Figure 1.1 shows an example of a typical gateway VoIP system-on-a-chip diagram.

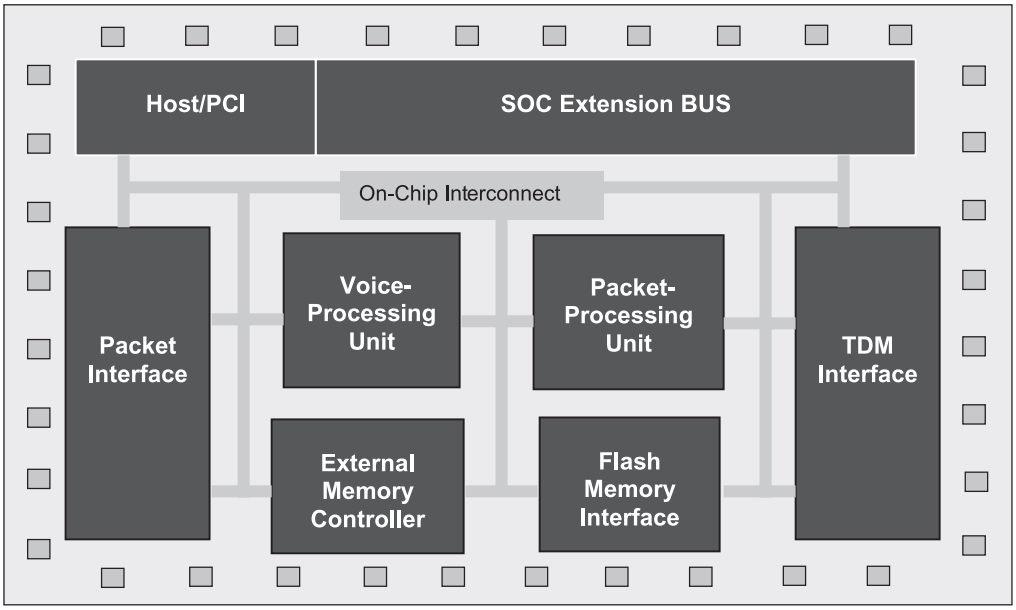

Fig. 1.1 A Typical Gateway SOC Architecture

1.2 VOICE OVER IP SOC

A gateway VoIP SOC is a device used for functions such as vocoders, echo cancellation, data/fax modems, and VoIP protocols. Currently, there are a number of these devices available from several vendors; typically these devices differ from each other by the type of functions and voice-processing algorithms they support.

In this example, we define the major blocks required to support carrier-class voice processing. The SOC can vary depending on the particular I/O and voice-processing requirements of the mediation gateway architecture. Major units for this SOC are as follows.

Host/PCI

The host interface is for control, code download, monitoring, and in some cases data transport. This host interface could be either a microprocessor-specific interface or a generic system-bus interface such as PCI.

- **Microprocessor Interface** A synchronous processor interface, such as a 32-bit synchronous Motorola 68000 or Intel 960 style interface operating at 33MHz with interrupt support, allows the SOC to interface to most processors with minimal glue logic. This interface usually supports multiplexed data and addresses to reduce the number of I/Os on the SOC. The SOC also supports interrupt generation in order to notify the CPU of external events.
- **PCI Interface** The SOC may have a PCI-compliant interface for communication with external processors and resources. The PCI interface would support bus Target (Slave) and Initiator (Master) functions and DMA, but would not require an arbiter. This interface also provides access to shared memory.

External Memory Controller

The external memory controller supports industry-standard inexpensive fast memory such as SDRAM. This memory is used to store code and data for processing elements within the SOC. Depending on the actual SOC architecture and fabrication process, the memory interface could require support for one 32-bit SDRAM module, two 16-bit modules operating at up to 133MHz.

Flash Memory Interface

A standard parallel flash port for access to boot programs, configuration data, and programs is available and accessible upon system reset.

Packet Interface

The packet interface can be Ethernet or Utopia.

- **Ethernet** A standard 10/100BT Ethernet MII or RMII interface may be useful in cases where both compression and packetization are performed in the SOC. In such architectures, IP packets may be transported within a system using Ethernet as the physical transport layer.
- **Utopia** An industry standard, Utopia level 2 interface is useful for interfacing to system fabrics that use ATM as a physical transport. This interface supports connections to ATM 155Mbit/s physical-layer interfaces.

TDM Interface

The TDM interface is the downstream interface to PSTN TDM streams. These are uncompressed voice channels of 64Kbit/s A-LAW/µ-LAW voice that is delivered to the SOC for compression and forwarding to the packet network. The SOC interfaces directly with legacy TDM device interfaces such as the ECTF H.100/H.110 standard serial interface.

- **ECTF H.100/H.110** H.100/H.110 is a standard TDM interface for legacy telephony equipment. H.100/H.110 allows the transport of up to 4096 simplex channels of voice or data on one connector or ribbon cable. This voice traffic may come from a WAN interface board, chip, or any other voice-processing device in the carrier systems described above. H.100 defines a mezzanine connection that can interface to other H.100 devices or to legacy MVIP/SCSA devices.

SOC Extension Bus

The SOC extension bus is required to load balance the system and to provide a unified host interface for access.

Voice/Tone Processing Unit

The voice/tone processing unit consists of multiple DSP cores that perform the following functions:

- Code excited linear prediction (CELP)
- Pulse code modulation (PCM)
- Echo cancellation
- Silence suppression
- Voice activity detector (VAD)
- Tone detection/generation
- Dual-tone multifrequency (DTMF)

Packet Processing Unit

The packet-processing unit consists of several packet processors that process the voice and signaling packets that are ready for transmission. This unit performs the following functions:

- ATM Adaptation Layer 1 (AAL1)
- ATM Adaptation Layer 2 (AAL2)
- User Datagram Protocol (UDP)
- Transfer Control Protocol (TCP)

We will spend more time on this gateway SOC in Chapter 3. Let's look at another SOC example. Figure 1.2 shows an overview diagram of a set-top-box (STB) SOC.

The major blocks in Figure 1.2 and their functions are listed below:

- Video processing unit (MPEG-2 codec)
- Digital signal processing (DSP) for AC3 audio processing
- CPU for control and transport of streams
- Modulation unit such as quadrature phase shift keying (QPSK) for satellite and quadrature amplitude modulation (QAM) for cable inputs

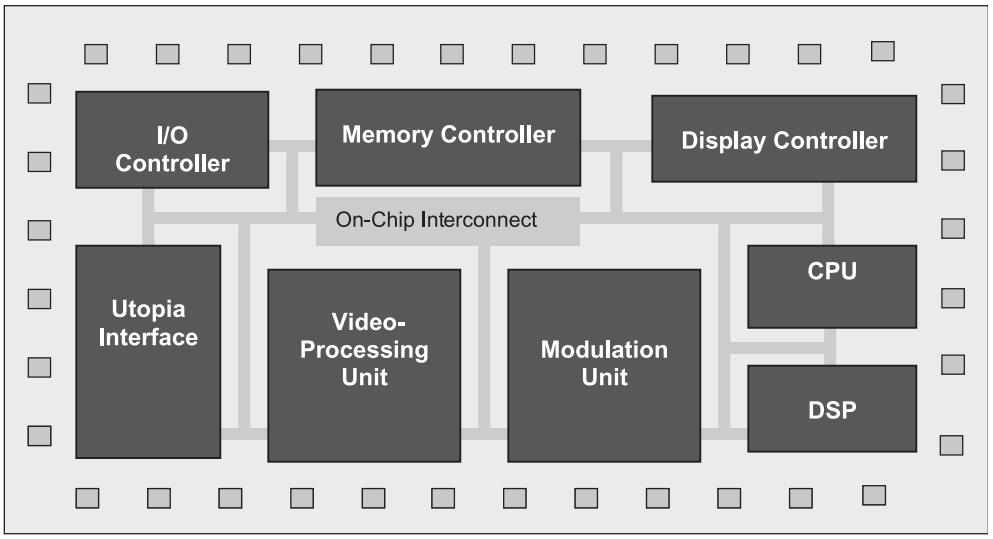

Fig. 1.2 Set-Top-Box SOC

- Utopia for cable modem interface
- Memory controller such as SDRAM controller
- I/O controller
- Display controller

A more detailed example of an STB is presented in Section 3.4.

In many SOC designs, you will find the following characteristics:

- Hierarchical architecture
- Hierarchical methods for physical design (placement and routing) and timing analysis
- On-chip interconnect
- Standard core-to-core communication protocols
- Hardware/Software codesign/verification
- Reusable infrastructure

Before we go further on SOC design, we need to introduce the concept of an IP.

1.3 INTELLECTUAL PROPERTY

In today's rapidly growing IC technology, the number of gates per chip can reach several millions, exceeding Moore's law: "The capacity of electronic circuits doubles every 18 months." To overcome the design gap generated by such fast-growing capacity and lack of available manpower, reuse of the existing designs becomes a vital concept in design methodology. IC designers typically use predesigned modules to avoid reinventing the wheel for every new product. Utilizing the predesigned modules accelerates the development of new products to meet today's time-to-market challenges. By practicing design-reuse techniques—that is, using blocks that have been designed, verified, and used previously—various blocks of a large ASIC/SOC can be assembled quite rapidly. Another advantage of reusing existing blocks is to reduce the possibility of failure based on design and verification of a block for the first time. These predesigned modules are commonly called Intellectual Property (IP) cores or Virtual Components (VC).

Designing an IP block generally requires greater effort and higher cost. However, due to its reusable architecture, once an IP is designed and verified, its reuse in future designs saves significant time and effort in the long run. Designers can either outsource these reusable blocks from third-party IP vendors or design them inhouse. Figure 1.3 represents an approximation of the amount of resources used in several designs with and without utilizing the design-reuse techniques.

As shown in Figure 1.3, the time and cost to design the first reusable block are higher than those for the design without reusability. However, as the number of usages increases, the time-saving and cost-saving benefits become apparent.

Licensing the IP cores from IP provider companies has become more popular in the electronic industry than designing inhouse reusable blocks for the following reasons:

1. Lack of expertise in designing application-specific reusable building blocks.

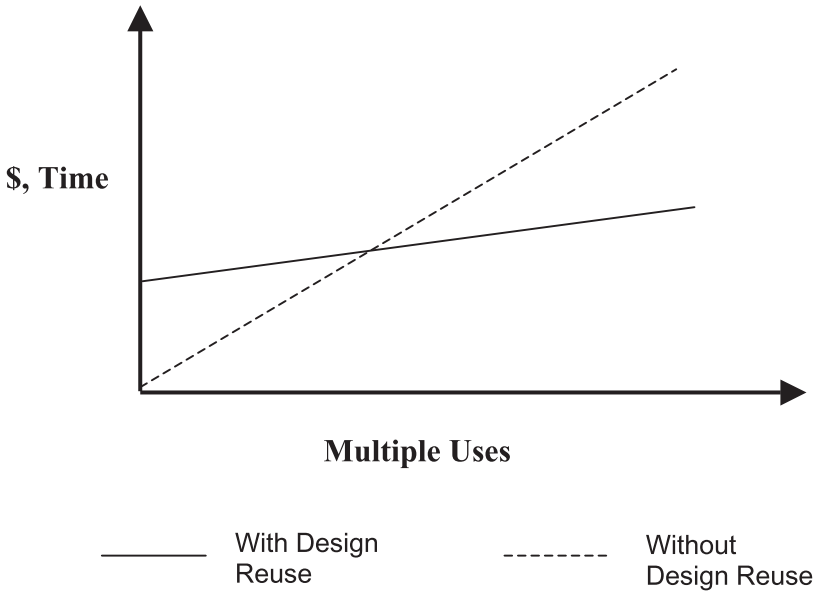

Fig. 1.3 Resources versus Number of Uses

2. Savings in time and cost to produce more complex designs when using third-party IP cores.
3. Ease of integration for available IP cores into more complicated systems.
4. Commercially available IP cores are preverified and reduce the design risk.
5. Significant improvement to the product design cycle.

Intellectual Property Categories

To provide various levels of flexibility for reuse and optimization, IP cores are classified into three distinct categories: hard, soft, and firm.

Hard IP cores consist of hard layouts using particular physical design libraries and are delivered in masked-level designed blocks (GDSII format). These cores offer optimized implementation and the highest performance for their chosen physical library. The inte-

gration of hard IP cores is quite simple and the core can be dropped into an SOC physical design with minor integration effort. However, hard cores are technology dependent and provide minimum flexibility and portability in reconfiguration and integration across multiple designs and technologies.

Soft IP cores are delivered as RTL VHDL/Verilog code to provide functional descriptions of IPs. These cores offer maximum flexibility and reconfigurability to match the requirements of a specific design application. Although soft cores provide the maximum flexibility for changing their features, they must be synthesized, optimized, and verified by their user before integration into designs. Some of these tasks could be performed by IP providers; however, it's not possible for the provider to support all the potential libraries. Therefore, the quality of a soft IP is highly dependent on the effort needed in the IP integration stage of SOC design.

Firm IP cores bring the best of both worlds and balance the high performance and optimization properties of hard IPs with the flexibility of soft IPs. These cores are delivered in the form of targeted netlists to specific physical libraries after going through synthesis without performing the physical layout. Figure 1.4 represents the role of firm IP cores in ASIC design flow.

In Figure 1.4, the tasks in shaded boxes can be covered by Firm IP and as a result accelerate the design flow. Table 1.1 provides a brief comparison of different IP formats.

Table 1.2 provides a collection of some of the deliverable items for different IP formats.

Table 1.1 Comparison of Different Intellectual Property Formats

IP Format	Representation	Optimization	Technology	Reusability
Hard	GDSII	Very High	Technology Dependent	Low
Soft	RTL	Low	Technology Independent	Very High
Firm	Targeted Netlist	High	Technology Generic	High

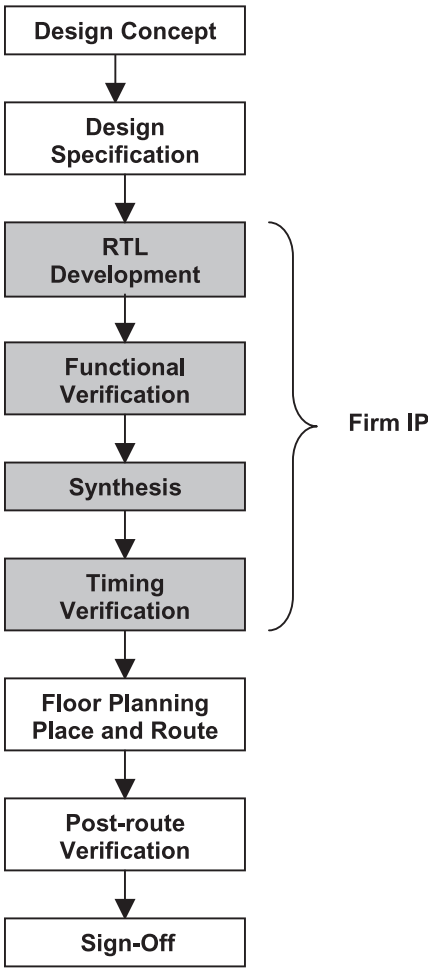

Fig. 1.4 ASIC Design Flow

Guidelines for Outsourcing IP

Although licensing IP can greatly enhance project design cycles, it can also hurt project schedules if the following are not carefully considered when selecting an IP vendor.

- ☞ Outsource IPs from a well-known IP provider with large customer base and great track record. Customer testimonials of integrating a specific IP from a third-party vendor represent

Table 1.2 Some of the Deliverables for Various IP Formats

Deliverables	Hard IP	Soft IP	Firm IP
HDL RTL code		•	
HDL targeted netlist			•
GDSII file	•		
Functional verification testbenches	•	•	•
Bus functional models	•	•	•
Floor planning models	•		
Synthesis and timing models	•	•	•
Full documentation	•	•	•

the best way of ensuring that the IP works in the integration process.

- Evaluate the IP functionality using demos and executable models before purchasing. Hardware demonstrations by IP providers are another way of ensuring that IP blocks are functional in silicon. Access to executable models allows you to change different parameters and make sure the IP provides functional results that you expect for your design.
- Ask for a full verification test environment. A full verification environment provides a set of models for different stimuli to verify the IP functionality and makes the overall chip verification less complicated.
- IPs should be accompanied by detailed documentation, such as datasheet, databook, user's guide, application notes, etc. Proper documentation offers valuable information on timing, interface definition, and different configurations for specific applications.
- Allocate a certain period of time to become familiar with the interfaces and functionality of the outsourced IP. It is quite common that IP interfaces do not match the rest of the system interface causing additional work to be done in the integration

process. This could change the project schedule if the additional integration time is not included in the project timeline.

- Make an agreement with the IP provider for technical support during the integration process. There are many instances when an IP has to be customized for a specific design at the integration time and only the IP provider is able to perform these modifications. Therefore, it is necessary to have the IP provider's support through the integration process.

We will cover more on IP verification and integration in Chapter 3. Table 1.3 shows several examples of Silicon IPs.

Table 1.3 Examples of IPs

Category	Intellectual Property
Processor	ARM7, ARM9, and ARM10, ARC
Application-Specific DSP	ADPCM, CELP, MPEG-2, MPEG-4, Turbo Code, Viterbi, Reed Solomon, AES
Mixed Signal	ADCs, DACs, Audio Codecs, PLLs, OpAmps, Analog MUX
I/Os	PCI, USB, 1394, 1284, E-IDE, IRDA
Miscellaneous	UARTs, DRAM Controller, Timers, Interrupt Controller, DMA Controller, SDRAM Controller, Flash Controller, Ethernet 10/100 MAC

1.4 SOC Design Challenges

Why does it take longer to design SOCs compared to traditional ASICs? To answer this question, we must examine factors influencing the degree of difficulty and Turn Around Time (TAT) for designing ASICs and SOCs. Usually for an ASIC, the following factors influence TAT:

- Frequency of the design
- Number of clock domains

- Number of gates
- Density
- Number of blocks

Another factor that influences TAT for SOCs is system integration (mainly integrating different silicon IPs on the same IC) that is one of the key factors in TAT. In a typical SOC, you deal with complex data flows and multiple cores such as CPUs, DSPs, DMA, and peripherals. Therefore, resource sharing becomes an issue. Figure 1.5 shows a bus-based approach to integration. Here, the architecture is tightly coupled, which is advantageous for performance, area,

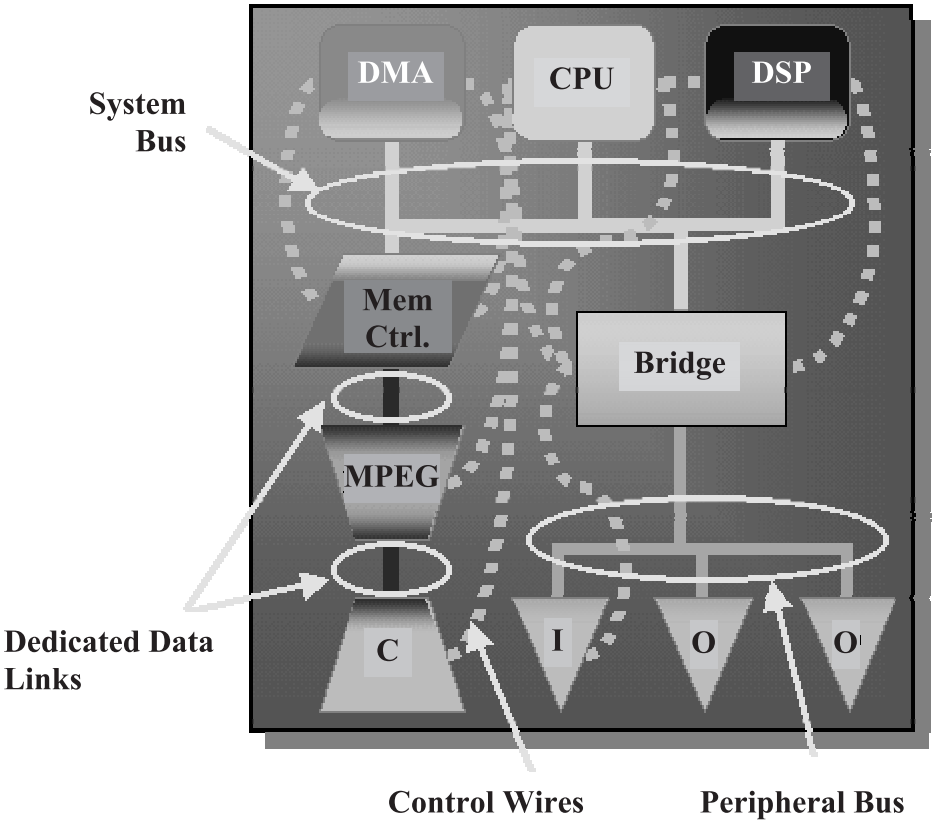

Fig. 1.5 A Traditional SOC Architecture (Copyright 2002, Sonics, Inc.)

and efficiency. However, communication between IPs becomes very complicated.

Let's examine this approach, as it is common practice among chip architects and designers. Here the CPU, DMA, and the DSP engine all share the same bus (the CPU or the system bus). Also, there are dedicated data links and a lot of control wires between blocks. Additionally, there are peripheral buses between subsystems. As a result, there is excessive interdependency between blocks and a lot of wires in the chip. Therefore, verification, test, and physical design all become difficult to fulfill.

A solution to this system integration is to use an intelligent, on-chip interconnect that unifies all the traffic into a single entity. An example of this is Sonics' SMART Interconnect SiliconBackplane MicroNetwork.

A MicroNetwork is a heterogeneous, integrated network that unifies, decouples, and manages all of the communication between processors, memories, and input/output devices. Figure 1.6 shows an SOC design using MicroNetwork architecture. An example of a MicroNetwork is Sonics' SiliconBackplane, which guarantees end-to-end performance by managing all communications among IP

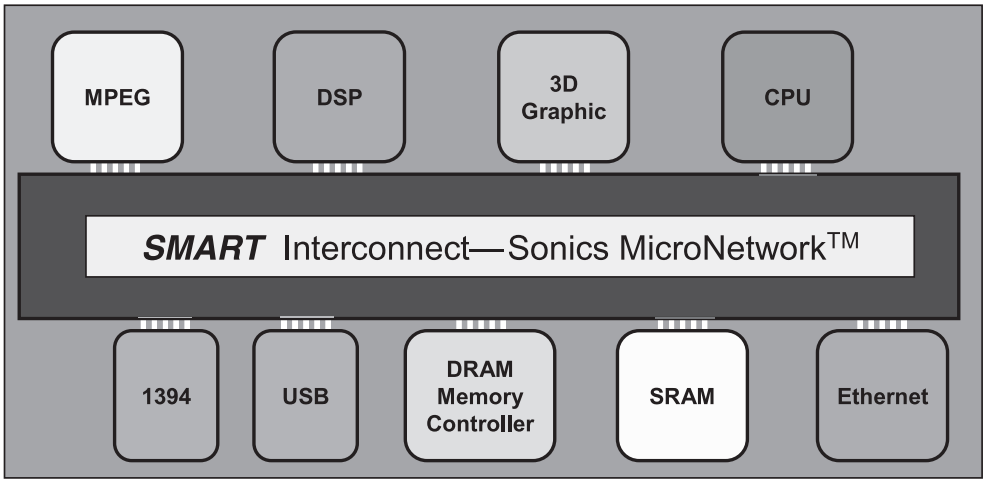

Fig. 1.6 Sonics' SiliconBackplane Used in SOC Design Architecture (Copyright 2002, Sonics, Inc.)

cores, as well as ensuring high-speed access to the shared memories common in typical SOC designs.

SiliconBackplane uses a standard core interface known as the Open Core Protocol (OCP), which delivers the first openly licensed, core-centric protocol. OCP comprehensively fulfills system-level integration requirements. The OCP defines a comprehensive, bus-independent, high-performance, and configurable interface between IP cores and on-chip communication subsystems. OCP is a functional superset of the Virtual Socket Interface (VSI) Alliance virtual-component-interface (VCI) specification, and enables SOC designers and semiconductor IP developers to prepare their cores for plug-and-play integration using Sonics' SiliconBackplane. Appendix B provides more information on OCP.

An SOC designer can optimize the design under development by optimizing the SiliconBackplane using a development environment developed by Sonics. Configuration and tuning parameters can be efficiently selected to optimize the SiliconBackplane and, as a result, to optimize the SOC design. The development environment consists of tools to wrap and package IP cores for integration as well as an automated basic configuration of the SiliconBackplane, and stimulus/performance analysis tools for successively refining SOCs.

When compared to a traditional CPU bus, an on-chip interconnect such as Sonics SiliconBackplane has the following advantages:

- Higher efficiency
- Flexible configuration
- Guaranteed bandwidth and latency
- Integrated arbitration

Design verification is another key challenge in designing SOCs. Verification has to happen at all levels of hierarchy, such as core/IP level, interface, and chip level. The integration of several cores on a single chip brings with it new challenges to the testing methodology even when the individual cores have design for test (DFT) already built in. The cores may have different types of testability: scan, built-in self-test (BIST), and functional. The integrator of the cores

must decide on a coherent test style from the outset and choose the cores accordingly. This, in turn, implies that the integrator has access to a number of IP providers and he or she has established an acceptance criterion for cores.

Chapter 3 covers the verification of cores and SOCs in more detail.

1.5 DESIGN METHODOLOGY

A good design methodology for ASICs and SOCs consists of a set of defined design flows for both front and back ends as well as tool integration and task automation. Let's start with the design flow. Figure 1.4 showed a typical top-down ASIC design flow. The flow can be divided into the following major parts: design entry, design implementation, design verification, physical design, and IC production. A more detailed front-end flow diagram is shown in Figure 1.7. Let's look at the steps involved in this flow.

The designer develops the RTL code that implements the functional specification. Chip designers should follow any coding guidelines provided by ASIC vendors.

Simulations at the register-transfer (RT) level should be thorough because this is really the only place where correct function can be verified efficiently. Simulations at the gate level are much too slow to be complete and static timing analysis (STA) does not verify functionality, only timing.

The synthesis tool generates both forward and backward annotation files. The forward annotation provides constraints to timing-driven layout tools while the back-annotated files provide delay information to either a simulator for gate-level simulations or a static timing analyzer.

The designer is responsible for verifying the synthesized gates for functional correctness and for estimated performance. Whether the verification is done with a simulator or a static timing analyzer, the wire loads are only estimates. The gate delays come from the

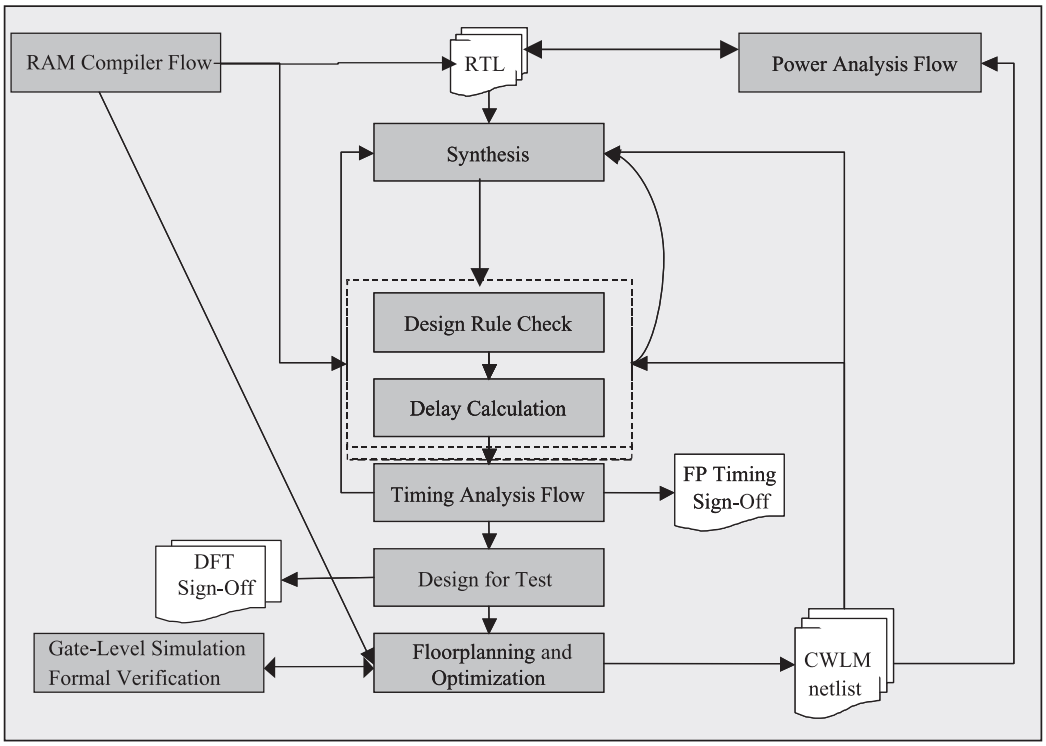

Fig. 1.7 A Front-End ASIC Design Flow (Printed with permission of Fujitsu Microelectronics America, Inc.)

technology library and are accurate. The delays are provided from the synthesis tool via a standard delay format (SDF) file.

Floorplanning takes information from the synthesis step to group the cells to meet the timing performance. It feeds back more accurate wire-load models to the synthesis tool and it provides the framework for place and route.

Figure 1.8 shows a spiral design flow. This type of flow is becoming popular with SOC designers for the front end. Here, the designers work simultaneously on each phase of the design until the design is gradually completed.

Once you finish the front-end work and generate a gate-level netlist for your design (ASIC or SOC), then you can start the physical design process.

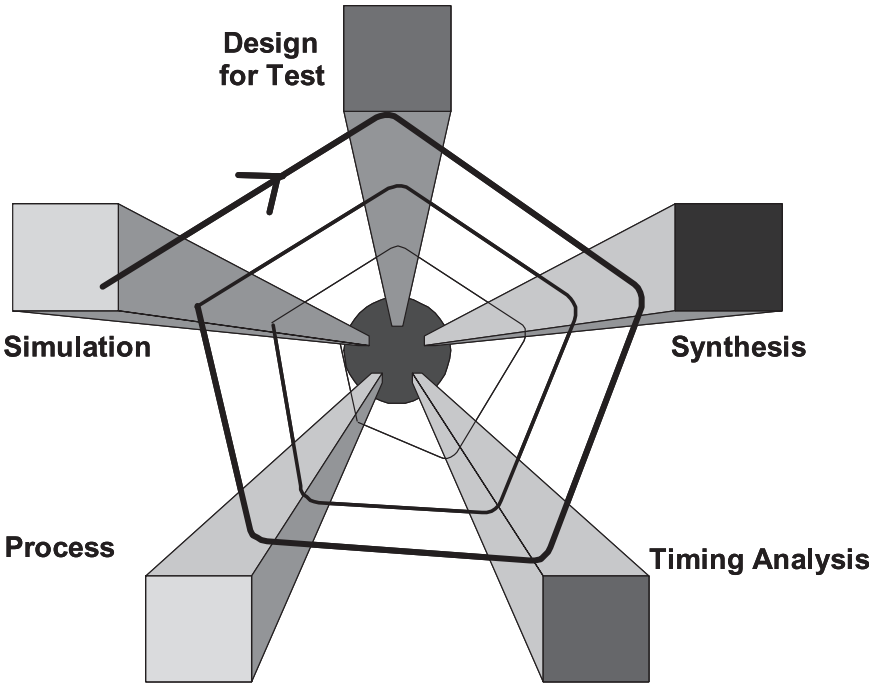

Fig. 1.8 Spiral Design Flow

Figure 1.9 shows a generic physical design, or back-end flow. The major steps consist of place and route, timing verification, and physical verification.

The inputs to place and route are netlist, clock definition, and I/O specification. The goal of place and route is to generate a GDSII file for tapeout. The place-and-route step performs placement, routing, clock-tree synthesis, optimization, and delay calculation.

Task automation is covered in Chapter 3.

1.6 SUMMARY

In this introductory chapter, we defined an SOC and some of its differences from a traditional ASIC. A key concept in SOC design is

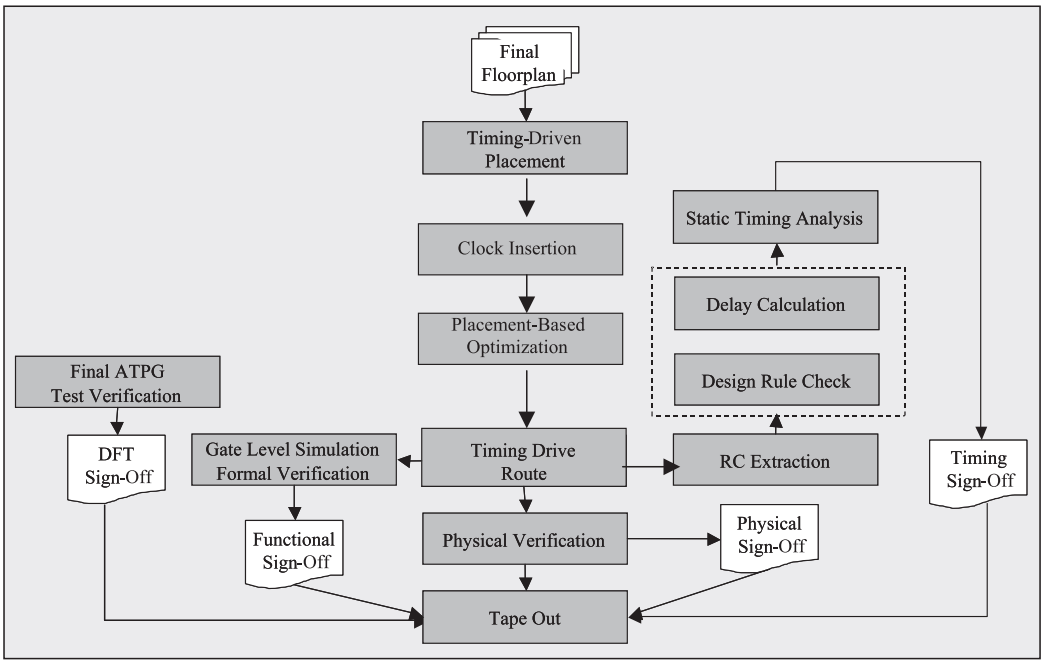

Fig. 1.9 Generic Physical Design Flow (Printed with permission of Fujitsu Microelectronics America, Inc.)

the usage of different IPs. This by itself creates a big challenge in SOC design, namely IP integration.

Reuse methodology is an important factor in SOC designs that reduces time-to-market (TTM). We cover more on ASICs and SOCs, including verification techniques, in Chapters 2 and 3, respectively.

Chapter 4 deals with the physical design domain that is common to both ASICs and SOCs. Once you have a netlist for the proposed IC (ASIC or SOC), then you enter the world of the physical domain.

Chapter 5 covers low-power design concepts and techniques that again are common to both ASICs and SOCs. Several methods of power optimization at different levels of abstraction will be covered. These techniques include algorithm, architecture, Register Transfer, and gate-level optimizations.

1.7 REFERENCES

1. M. Keating and P. Bricaud. *Reuse Methodology Manual for System-on-a-Chip Designs*. Norwell, MA: Kluwer Academic Publishers, 1998.
2. F. Nekoogar. *Timing Verification of Application-Specific Integrated Circuits (ASICs)*. Upper Saddle River, NJ: Prentice Hall PTR, 1999.
3. P. Rashinkar, P. Paterson, and L. Singh. *System-on-a-Chip Verification Methodology and Techniques*. Norwell, MA: Kluwer Academic Publishers, 2001.
4. H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd. *Surviving the SOC Revolution: A Guide to Platform-Based Designs*. Norwell, MA: Kluwer Academic Publishers, July 1999.
5. S. Azimi. *Overcoming Challenges and Obstacles to System on Chip (SOC) Products*. Sunnyvale, CA: Marvell Semiconductor, Inc., 2000.
6. A. Qureshi (Cadence Design Systems, Inc.). "SOC Design Methodology and Ideal Structures." DesignCon2000.
7. D. Wingard. "Integrating Semiconductor IP Using microNetworks, ASIC Design." Mountain View, CA: Sonics, Inc., 2001.
8. R. Fehr. "Intellectual Property: A Solution for System Design." Technology Leadership Day, October 2000.
9. P. Levin and R. Ludwig. "Crossroads for Mixed-Signal Chips." *IEEE Spectrum*, March 2002.
10. R. Rajsuman, *System-on-a-Chip Design and Test*. Santa Clara, CA: Artech House Publishers, 2000.