

Introduction to ... The Design of Mixed-Signal Systems on Chip¹

Ken Kundert
Cadence Design Systems

Design of Mixed-Signal Systems on Chip
35th Design Automation Conference, 1998

DnC

Henry Chang	Felicia James
Dan Jefferies	Ken Kundert
Lee Stoian	Richard Trihy

Slide 1

Preview

Introduction
◆ Ken Kundert — Cadence Design Systems

Issues in Mixed-Signal System-on-Chip Design
◆ Felicia James — Texas Instruments

Implementing Top-Down Mixed-Signal Design
◆ Dan Jefferies — Cadence Design Systems

An Introduction to Verilog-AMS and VHDL-AMS
◆ Richard Trihy — Cadence Design Systems

Mixed-Signal Virtual Socket Interface
◆ Henry Chang — Cadence Design Systems

Reuse and Exchange of Mixed-Signal Blocks
◆ Lee Stoian — SiPCore

March 26, 1998

DnC

Slide 2

Abstract

The electronics industry is increasingly focused on the consumer marketplace, which requires low-cost high-volume products to be developed very rapidly. This, combined with advances in deep submicron technology has resulted in the ability and the need to put entire systems on a single chip. As more of the system is included on a single chip, it is increasingly likely that the chip will contain both analog and digital sections. This tutorial presents the issues involved in developing large single chip mixed-signal systems both from the design and CAD perspectives.

Target Audience

The primary audience is circuit designers and CAD engineers involved in the design of large or complex mixed-signal systems on chip.

Preview

I will start the tutorial by describing the two basic types of mixed-signal systems on chip (MS-SOC), custom and ASIC, and introduce some of the issues involved in developing them. Felicia James then presents the issues in greater depth and illustrates them with real examples. Dan Jefferies presents a top-down mixed-signal methodology that has proven to be successful on custom MS-SOC. Richard Trihy presents an overview of the Verilog-AMS and VHDL-AMS languages that will play a key role in the design of MS-SOC. Henry Chang describes the industry-wide effort to formalize the MS-SOC design processes that will be used to support the sharing of MS intellectual property (IP). Finally, Lee Stoian will present the issues in authoring mixed-signal IP for reuse, and the issues involved in integrating that IP on a MS-SOC.

¹ Presented during the “Design of Mixed-Signal Systems on Chip” short course at the 35th Design Automation Conference in 1998.

Glossary

- A/d** Also referred to as *big A, little d* mixed signal. See *custom MS-SOC*.
- AHDL** An analog hardware description language such as Verilog-A. Contrast with *MS-HDL*.
- AMS** Short for analog and mixed-signal. Generally used to refer to the new standard mixed-signal hardware description languages, Verilog-AMS and VHDL-AMS.
- ASIC MS-SOC** A mixed-signal circuit with a strong digital focus. Generally a large digital chip with a small number of mixed-signal interface blocks designed by a system expert. Contrast with *custom MS-SOC*.
- BBD** Block-based design, a system-on-chip design methodology. Contrast with *MS-TDD* and *SOC*.
- Custom MS-SOC** A mixed-signal circuit with a strong analog focus. Generally a high-performance mixed-signal chip designed by a mixed-signal expert. Contrast with *ASIC MS-SOC*.
- D/a** Also referred to as *Big D, little a* mixed signal. See *ASIC MS-SOC*.
- IP** Intellectual property. A block of circuitry that is packaged and sold for use in a system-on-a-chip.
- MS** Mixed-signal. A combination of analog and digital.
- MS-HDL** A mixed-signal hardware description language such as Verilog-AMS and VHDL-AMS.
- MS-SOC** A mixed-signal system on chip.
- MS-TDD** Mixed-signal top-down design. A mixed-signal system-on-chip design methodology. Contrast with *BBD* and *SOC*.
- SOC** System on chip. Either a complete system constructed by assembling blocks on a chip or a particular design methodology used to develop a system on a chip.
- TDD** A system-on-chip design methodology. When referring to mixed-signal circuits this means *top-down design* (see *MS-TDD*). When referring to digital circuits, it means *timing-driven design*.
- V*-AMS** A label that refers both to Verilog-AMS and VHDL-AMS mixed-signal hardware description languages.
- VC** A virtual component. A block of circuitry that has been designed for reuse by following VSI guidelines.
- VSI** The virtual socket interface. A collection of open guidelines and standards proposed by the VSI Alliance to promote IP exchange.

Global Market Trends

Electronics Market is Reaching \$1T


- ◆ Soon to become the largest industrial sector

Electronics is Becoming a Consumer Marketplace

- ◆ Products lifetime is measured in months
 - Time to market pressure is intense
- ◆ Cost constraints are rigid
 - Systems implemented in silicon to reduce costs

Complexity and Size of Circuits Continues to Increase

- ◆ CAD is not keeping up, requiring changes in design methodology

March 26, 1998 3 

Slide 3

Global Market Trends

The electronics market is reaching \$1T and it is soon expected to eclipse transportation and become the largest industrial sector in the world. It has done that by becoming a consumer marketplace. It has to; nothing else can support a market that large. Consumer markets are faster paced than either industrial or military markets. Product lifetimes are very short, because there are usually many competitors and because consumers tend to be fashion conscious and faddish. In a consumer market, time to market pressure is intense and cost constraints are rigid.

Market Drivers

	1997-0.35 μ	1998-0.25 μ	1999-0.18 μ
Silicon Complexity	300 Kgates	1 M gates	5 M gates
Initial Design Cycle	14 months	10 months	8 months
Derivative Cycle	7 months	5 months	2 months
Primary IP Sources	Intra-Group	Inter-Group	Inter-Company

- Costs force systems to move from board to chip
- System is too complex for any one group to design
- Acquire blocks externally to reduce complexity

March 27, 1998 4

DAC

Slide 4

Market Drivers

The fast pace of the consumer market and the huge number of transistors becoming available on chip is causing the implementation of systems to shift from being on board to being on chip. When implemented on a board, the system designer assembles existing blocks into a system. This same paradigm must be supported on chip in order to quickly and efficiently design systems that are low cost, low power, and light weight, as required by the market. However, the greater number constraints on chip make that difficult.

Out-Sourcing Design

SOC Becoming Too Large for Any One Organization

- Too many specialties embodied in a typical system
- Design centers with unique specialties will work together
- Two types of design groups
 - Those that author the blocks
 - Those that assemble the blocks

IP Authors

- Must design for reuse

IP Assemblers

- Must verify that whole system together as expected

March 26, 1998 5

DAC

Slide 5

Out-Sourcing Design

With the move to high volume consumer markets, systems are increasingly implemented in silicon to reduce costs. The complexity and size of circuits continues to increase. No one organization can hope to have all of the specialized knowledge needed to produce a competitive complex system in silicon. The

electronics industry is responding to these changes by becoming more specialized and de-verticalized. In the future several companies with unique specialties will work together to produce MS system chips.

This process is just beginning in design. Cadence's Design Factories are an excellent example. Initially, design out-sourcing was used by those with need but little ability. As systems become more complex, and the blocks that make up the system require more specialized knowledge, more companies are taking advantage of design out-sourcing to get access to expertise that they do not have internally.

Design out-sourcing requires two critical ingredients, available intellectual property (IP) and the ability to quickly and reliably assemble the IP into a system on silicon. The challenge for the CAD industry is to provide support for portable IP and methodologies that allow distributed hierarchical design. The industry is pursuing the Virtual Socket Interface (VSI) as a way of addressing these needs.

Importance of Mixed-Signal

Larger SOCs Are Increasingly Mixed-Signal

- Most systems include some analog
- The more of the system included on chip, the more likely its mixed-signal

April 16, 1998 6

DAC

Slide 6

Importance of Mixed Signal

As more of the system is included on a signal chip, the more likely the chip will be mixed signal, because most systems include at least some analog or mixed-signal at the interfaces. Examples include multimedia systems such as DVD, graphics systems, magnetic disk drives, digital video cameras and set-top boxes. Wireless applications include cellular phones, wireless LANs, and etc.

Mixed-signal circuits are subject to the same trends as electronics as a whole: increasing complexity and shrinking time to market. However, in an important way, these issues are even more of a concern in the mixed-signal area because of the lower levels of design

automation (for example, the lack of analog synthesis and test) and the shortage of skilled mixed-signal designers. Thus, it is critical to improve designer productivity, however doing so with design reuse and portable IP is more difficult than with digital circuits.

Design Challenges

- Increasing Complexity*
- Decreasing Time to Market*
- Reuse*

April 16, 1998 7 **DnC**

Slide 7

Design Challenges

The three main issues that are confronting mixed-signal designers today are

1. The complexity of the circuits they are expected to design is growing rapidly.
2. The time available to design a product is shrinking because of competitive pressures.
3. One way of addressing the first two issues is to reuse existing blocks. However, this brings its own issues.

Increasing Complexity

- Increasing Complexity as Circuits become Larger*
 - ◆ Increasing Integration
 - To reduce cost, size, weight, and power dissipation
- Increasing Complexity of Signal Processing*
 - ◆ Implementation of algorithms in silicon
 - Adaptive circuits, error correction, PLL's, etc.
 - ◆ Digitalization
 - Both digital information and digital implementation

April 16, 1998 8 **DnC**

Slide 8

Increasing Complexity

There are several trends that tend to increase the complexity of mixed-signal circuits. First the number of transistors available to implement a system on chip is increasing with advances in process technology. These transistors are used to implement more of the system, or more sophisticated systems, on chip, thereby decreasing cost, size, weight, and power while increasing functionality.

Second, the move from implementing cells to blocks to systems means that the things being designed look less like simple functional blocks and more like algorithms. For example, ADCs have moved from simple flash converters, to pipelined converters, to $\Delta\Sigma$ converters. In addition, there is increasing use auto-calibration, error correction and adaptive filtering. Finally, the increasing use of both digital forms of signals, such as in digital wireless communications, and digital implementations, increases complexity.

Decreasing Time to Market


First to Market Captures Majority of Market

Stiff competition

- ◆ More competitors

Shrinking product lifetime

- ◆ Customers are more fashion conscious, faddish

March 31, 1998 9 

Slide 9

Decreasing Time to Market

The consumer marketplace tends to have many competitors, which implies substantial time-to-market pressure.

1. The first to market with a new idea or capability generally captures the majority of the market.
2. Designs must be brought to market quickly to avoid being out-of-date when they reach the market.
3. Product lifetimes are generally very short.

Currently an extreme example of this situation is occurring in the digital camera market, where the product life cycle is often only 2-3 months.

Example: Disk Channel Chip Market

Market Moving from Preamps to Channel Chips


- ◆ From hundreds to tens of thousands of transistors
 - From functional block to "algorithm in hardware"

Being First to Market is Essential

- ◆ No second sources
 - "First company to fill the socket wins"
- ◆ From samples to 1M units/month run rate in 2 months
- ◆ Production runs last 8 months, then obsolescence

Market is Intolerant of Mistakes

- ◆ Need for additional design turn can kill product
- ◆ High reliability essential
 - Disk failure is catastrophic for end user

March 27, 1998 10 

Slide 10

Disk Channel Chip Market

An example that shows the increase in complexity and time-to-market pressures is found in the disk drive

channel chip market. Recently, a particular company in this market segment was forced by competitive pressures to move from making pre-amps to making a single chip containing the whole channel. The pre-amp is a simple functional block implemented with perhaps a hundred transistors, whereas a channel chip implements a complex mixed-signal algorithm in hardware and requires tens of thousands of transistors.

In this market, there are no second sources, so the first vendor to fill the socket wins the business and the rest go home. Multiple design iterations are not an option. Once a working prototype exists and has been accepted, the chip manufacturer is expected to move to full production of over 1M parts per month in two months. Production typically lasts 6-8 months, at which point the product is out-of-date and production stops. Because of the rapid pace of the process, vendors live and die by their reputation. If a vendor delays production due to a design flaw, or builds unreliable parts, they are not invited to bid for the next product.

Reuse


Rapid Assembly of Systems from Existing Blocks

Why Do It?

- ◆ Reduce complexity of design by limiting it to system level
- ◆ Reduce time to market
- ◆ Supports specialization in design skills

Challenges

- ◆ Blocks must be designed and packaged for reuse
 - Robust interfaces
 - Specifications and application notes
 - High level models
 - Physical design data
- ◆ Blocks must be assembled and verified

March 27, 1998 11 

Slide 11

Reuse

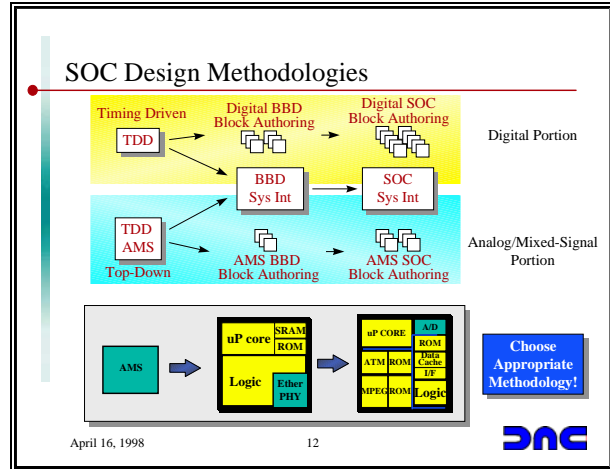
Reuse refers to the practice of rapidly assembling systems from existing blocks. The fundamental idea is to reduce the cost of designing a block by amortizing it over several systems. Reusing existing blocks can also dramatically reduce the time required to complete the design.

The issues involved in reusing existing blocks also surface when using externally designed blocks, regardless of whether they were meant to be reused. One would use externally designed blocks to get access to blocks that you do not have the resources or the skill to design. We use reuse to illustrate the issues that

occur when blocks are designed externally, are pre-specified, and are difficult if not impossible to change.

Reuse requires that system designers be able to determine if a block meets the needs of the system and operates properly in the system. The block must have a robust interface and must be carefully documented. The documentation must include high level models and physical dimensions. In addition, the system engineers are generally not experts in how to apply the blocks, and so require application support and application notes.

Reuse has not really caught on for mixed-signal design because mixed-signal circuits are very sensitive to the parameters of the process they are implemented with and difficult to retarget to new processes. This leads to a short life for the design of a particular block. It is also difficult and time-consuming to generate the needed level of documentation, particularly the high-level models. Currently, economics does not favor reuse for mixed-signal circuits, but future advances in the automation of re-targeting, documentation, and modeling of mixed-signal blocks will likely change this situation.



Slide 12

SOC Design Methodologies

Different design methodologies (see Table 1) are used depending various factors, such as the performance and time-to-market requirements.

High performance typically requires custom design, which if mixed-signal requires a top-down design (MS-TDD) methodology, where each of the cells in the design is custom designed for the system. The top-down design methodology is the only suitable approach when there is close coupling and complex interaction between the blocks, such as PRML channel chips or RF systems.

Designers typically use a block-based design (BBD) methodology when preexisting blocks are available, but have not been designed for reuse and so their interfaces must be customized for the system. BBD brings faster time-to-market and generally a broader set of features, but at the expense of performance.

	MS-TDD	BBD	SOC
Differentiator	Analog Performance	New Product Features	Complete set of features
Analog in System?	Yes, for Function & Performance	Only if benefits	Try for A/D + SW
Technology	Any	< 0.35 μm CMOS	< 0.35 μm CMOS
Example Designs	PRML, xDSL, RF	Block: RAMDAC Chip: Graphics Control	VC: A/D+SW Modem Chip: Set Top Box
Primary Design	Custom Logic / AMS	Blocks in context, custom interfaces	Interfacing to system and bus
Author / Integration?	Combined	Blocks for system	Block pre-verified, 'simple' I/O
Re-Use	Personal	Source and Core: hard	VC: hard, firm
Designers	AMS Designers	Block: AMS Designers Chip: Dig/AMS Des.	VC: AMS designers Chip: system designer
Design Focus	AMS	Blocks in context, custom interfaces	Interfacing to sys & bus

Table 1

Finally, a system-on-chip (SOC) methodology is used when virtual components (VC) are available. Virtual components are preexisting blocks that have been designed for reuse by conforming to VSI interface standards. In this case, the blocks are simply assembled on chip and routed using standard busses. The SOC methodology tends to result in fully featured and relatively flexible systems at the expense of performance.

MS-TDD designs tend to be primarily mixed-signal, whereas BBD and SOC tend to be largely digital designs with a small number of mixed-signal blocks.

Top-Down AMS Design

Design Hierarchically

- ◆ Verify specifications at the system level
- ◆ Design and verify system before designing blocks
- ◆ Design and verify blocks before designing cells


Supports Concurrent Design

- ◆ Once system is specified, blocks can be designed in parallel

Dramatically Speeds Simulation

- ◆ System simulation is executable spec for block designers
- ◆ Simulate block in context of system
- ◆ Requires AMS language simulator
 - Verilog-AMS
 - VHDL-AMS

March 31, 1998 13



Slide 13

Top-Down AMS Design

Top-down design is suitable for high-performance mixed-signal systems, especially those with a complex interaction between the analog and digital sections of the design. Excellent examples of systems that are best designed with a top-down design style include PRML-based disk channel chips and RF transceiver front-ends.

The basic principle of top-down design is to design and verify the system at an abstract level before beginning design at the next level down. It is appropriate whenever there is sufficient complexity at the system level, in which case employing top-down design reduces the chance that blocks will have to be redesigned because they were originally designed with incorrect assumptions. It also naturally supports concurrent design once the system has been specified because the block designers can work relatively autonomously.

Top-down design also provides important benefits when verifying the functionality and performance of a system with simulation. When designing the system,

the blocks are described with a behavioral language and the system is verified at an abstract level. Thus, top-down design of mixed-signal systems requires a mixed-signal hardware description language (MS-HDL), such as Verilog-AMS or VHDL-AMS. Abstract models generally reduce simulation time several orders of magnitude compared with circuit level simulation.

If the system- and block-level simulation is performed with the same language, then the MS-HDL description of the system is very important to the block designer. The behavioral model of the block represents an executable specification and the description of the system represents an executable test bench for the block.

Mixed-Level Simulation


Only Feasible Method to Verify Complex MS Systems

- ◆ Verify System Using Behavioral Description for Every Block
- ◆ Replace One Block at a Time With Transistor Level Netlist and Reverify
- ◆ Verifies Block at Transistor Level in Context of Full System
- ◆ Verifies Compatibility of Interfaces

Requires

- ◆ MS-HDL
- ◆ High Capacity Transistor Level Simulation

March 31, 1998 14



Slide 14

Mixed-Level Simulation

Mixed-level simulation is used during top-down design to verify large complex mixed-signal systems, and it is the only feasible approach currently available. Some propose to use either timing simulators (sometimes referred to as fast or reduced accuracy circuit simulators) or real circuit simulators running on parallel processors. However, both approaches defer system-level verification until the whole system is available at transistor level, and neither provides the performance nor the generality needed to verify most mixed-signal systems.

In mixed-level simulation, the system, described at a high level, acts as a test-bench for the block, which is described at the transistor level. Thus, the block is verified in the context of the system, and it is easy to see the effect of imperfections in the block on the performance of the system.

Mixed-level simulation verifies the functionality of each block and the interfaces between the blocks, but it

does not guarantee the system as a whole meets its performance specifications because the whole system is never simulated with each block simultaneously modeled at a low level. Thus, mixed-level simulation assures that the imperfections of each block individually does not compromise the performance of the system, but there is no assurance that the imperfections do not combine to collectively compromise the system's performance.

Case Study: Disk Read Channel

Impossible to Simulate at Circuit Level

- ◆ >10,000 transistors
- ◆ 2000 cycles needed to train adaptive circuits
- ◆ Predicted simulation time > 1 month

Impossible to Simulate Blocks Individually

- ◆ System involved complex feedback loop
- ◆ Unable to predict closed-loop performance from measurements on individual blocks
- ◆ Difficult to verify blocks outside feedback loop

Mixed-Level Simulation Was Only Feasible Approach

- ◆ 1 day for 2000 cycles with one block at circuit level

DAC

March 30, 1998 15

Slide 15

Case Study: Disk Read Channel

In an example that is now several years old, designers tried to simulate a PRML disk read channel chip at the circuit level. The circuit included over 10K transistors in analog blocks and about an equivalent number of gates described with a Verilog netlist. The PRML algorithm involves adaptive filtering and would require 2000 cycles to train the adaptive circuits, which implies that no meaningful measurements can be made until 2000 cycles have been simulated. The circuit is constructed with complex feedback loops that made it difficult to simulate the blocks alone.

The estimated simulation time was greater than 1 month and so deemed unfeasible. Timing simulators really only accelerate transistor-level simulation of digital circuitry, but the digital circuitry was being simulated by Verilog, so timing simulators offered no benefit. Spice-level simulation on parallel processors only offered a 3-4x speed-up, which took the month long simulations to a week, which was still too long.

Instead, the system was described with a combination of AHDL and a Verilog netlist. Simulation time for the system dropped to 10 minutes. When the largest block was simulated at the transistor level with the remainder of the system described at a high level, the simulation time was 1 day.

Bottom-Up Verification

System-Level Verification with Transistor-Level Effects

- ◆ Model behavior of block as implemented with behavioral models
- ◆ Remove implementation detail, keep behavior
- ◆ Simulate whole system using detailed behavioral models
- ◆ Captures performance problems due to interactions of blocks

Necessary for Reuse, BBD and SOC

- ◆ Allows rapid validation of system with model of block as implemented
- ◆ Hides implementation details of blocks

Rarely Done Today

DAC

March 30, 1998 16

Slide 16

Bottom-Up Verification

Bottom-up verification is the act of trying to predict the performance of the system by performing simulation of the whole system with each block being accurately described with a behavioral model. It differs from high-level simulation in that the behavioral model for each block has been supplemented so that it models the details of the behavior of the block as implemented. Thus, the block must be implemented and some type of extraction process used to generate the model. When done properly, it allows the detailed verification of very large systems. The behavioral simulation runs quickly because the details of the implementation are discarded while keeping the details of the behavior. Because the details of the implementation are discarded, the detailed behavioral models generated in a bottom-up verification process are useful for third-party IP evaluation.


Bottom-Up Verification is Rarely Done Today

Too Hard

- ◆ Can be as hard as designing block
- ◆ Designers are not modelers
- ◆ No automated tools or methodologies

A Barrier for BBD and SOC

- ◆ Without BUV, BBD and SOC are severely handicapped

March 31, 1998 17 

Slide 17

Bottom-Up Verification is Rarely Done Today

Though bottom-up verification is necessary to completely verify the performance of large systems, and is also necessary to the block-based and system-on-chip design styles, it is rarely done today. Generating behavioral models that include the detailed behavior of even simple blocks is quite difficult and requires a specialized background not commonly found in the design team. This situation is not expected to change until automated tools and methodologies develop to generate detailed behavioral models.

Mixed-level simulation is currently the best approach to verifying large mixed-signal systems that are designed with a top-down methodology. However, eventually systems will be too large to completely verify with mixed-level simulation, in which case a bottom-up verification approach will be necessary.


Custom and ASIC MS-SOC

Custom (A/d)

- ◆ Tends to be smaller with complex interaction between analog and digital sections. Designed by MS expert
- ◆ Examples
 - PRML Channel Chips

ASIC (D/a)

- ◆ Tends to be large digital chips with isolated analog sections with simple interaction between analog and digital sections. Designed by systems expert with little expertise in MS design.
- ◆ Examples
 - Multimedia processors

March 31, 1998 18 

Slide 18

Custom and ASIC MS-SOC

Mixed-signal systems-on-chip can generally be partitioned into two different types, custom and ASIC. These are often referred to as A/d (big A, little d) and D/a (big D, little a) mixed-signal. It is generally assumed that A/d has more analog transistors than digital gates, whereas D/a has more gates than transistors. However, the ratio of transistors to gates is not really the issue. Rather, it is the design style used, thus I use the *custom* and *ASIC* labels.

A custom MS-SOC is characterized by a complex interface between the analog/mixed-signal and digital sections and typically has a large percentage of analog content. They are generally high-performance oriented, are designed by analog or mixed-signal designers, and have relatively inflexible architectures. Custom MS-SOC tend to be smaller (up to several hundred thousand transistors) than ASIC MS-SOC and are designed with a top-down design style. Examples include disk channel chips and RF front-ends.

An ASIC MS-SOC can usually be described as a big digital chip (tens of millions of transistors) with a small number of relatively isolated mixed-signal interface blocks. It tends to be considerably larger than a custom MS-SOC but there is a much simpler interface between the digital and analog/mixed-signal sections. ASIC MS-SOCs are typically designed by system experts that have little knowledge of analog or mixed-signal design issues.

MS-SOC Issues

Parasitic Coupling

- ◆ Routing
- ◆ Layout
- ◆ Substrate
- ◆ Supplies
- ◆ Package

Capacity


- ◆ Systems on chip are large
 - They stress existing tools to breaking point

Test

- ◆ How to test analog?
 - Analog is 10-20% of circuitry
 - Analog requires 70% of test development time
 - Analog test is 50% of production costs
- ◆ How to avoid breaking digital test?

March 31, 1998

19



Slide 19

MS-SOC Issues

There are three main issues that are shared with all MS systems on chip: parasitic coupling, capacity, and test.

Parasitic coupling occurs when analog circuitry is sensitive to that the noise generated in digital circuitry. The noise couples into sensitive analog circuitry through many different paths. For example, noise capacitively couples into sensitive analog blocks or routing if they are placed too close to digital circuitry. However, even if placed far from each other, digital noise can couple into analog circuitry through the substrate, supplies, or package.

Full-chip systems are large, and can stress any tool or tool suite to the breaking point.

Finally, a very serious issue is test. Generally, digital test is far advanced over analog test, being much more efficient and complete. Though analog only represents 10-20% of the transistors in a typical mixed-signal chip, it can require 70% of the test development time and analog test represents over half of the production cost, a figure which continues to rise. In addition, if not carefully designed, the addition of analog circuitry can make it difficult to test the digital portion as well.


Issues Unique to Custom MS-SOC

Complex Interaction Between Analog and Digital

- ◆ Requires co-design
- ◆ Requires co-simulation
 - V*-AMS languages
- ◆ Requires true mixed-signal CAD system
 - Familiar and comfortable for digital designer
 - Familiar and comfortable for analog designer
 - True mixed-signal capabilities
 - MS back annotation of parasitics
 - MS HDL

March 31, 1998

20



Slide 20

Issues Unique to Custom MS-SOC

The issues that are unique to custom MS-SOCs center around their high-performance nature and the complex interaction between analog and digital circuitry. The analog and digital circuitry must be designed and simulated together. The CAD system used to design MS-SOC must be comfortable for both analog and digital designers. It must be able to easily import synthesized digital netlists, it must provide an MS-HDL, and it must support automatic back annotation of mixed-signal parasitics (resistance and capacitance for analog circuitry and delays for digital circuitry).

Issues Unique to ASIC MS-SOC

System Designer Unfamiliar with MS Issues and Needs

- ◆ Unrealistic specifications on mixed-signal sections
- ◆ Inadvertent coupling between analog and digital sections

Risk


- ◆ Analog circuitry is high risk
- ◆ Could cause additional design turns

Bottom-Up Verification

- ◆ Needed to verify full system

March 31, 1998

21



Slide 21

Issues Unique to ASIC MS-SOC

Typically, an ASIC MS-SOC is assembled by a system engineer who is very familiar with the application area but is not well versed on the care and feeding of analog and mixed-signal circuitry. This may result in them placing unrealistic expectations on the mixed-signal

blocks when writing their specifications, and it may result in inadvertent coupling between the analog and mixed-signal sections because mixed-signal design practices are not followed appropriately.

The analog and mixed-signal sections can add considerable risk to a ASIC MS-SOC. Problems in the mixed-signal circuitry can cause several additional design turns that would not be needed in a purely digital SOC.

Finally, during the design of an ASIC MS-SOC, bottom-up verification is needed to allow the whole system to be verified. However, bottom-up verification is not yet very practical.

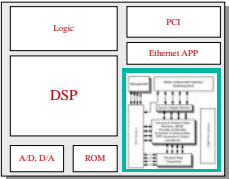
Block-Based Design Methodology


System Constructed from Existing Blocks

- ◆ Blocks not designed as Virtual Components (VC)
 - VCs have interfaces designed to standard busses for reuse
- ◆ Interface of blocks customized to fit in system

Blocks Not Designed for Reuse

- ◆ Not robust against changes in surroundings
- ◆ Possibly designed with incompatible design tools



March 31, 1998
22


Slide 22

Block-Based Design Methodology

The block-based design methodology involves assembling a collection of preexisting blocks into an MS-SOC. The blocks were not designed for reuse, and so the interface of each block must be modified in order to be integrated with the other blocks. Block-based design is used to construct systems that are larger than is possible with an MS top-down design style, but they also tend to be lower performance.

The challenges of a block-based design style stem from the blocks not being designed to work together. For example, the blocks are often not designed to be placed within an arbitrary system and are often designed using different CAD tool flows.

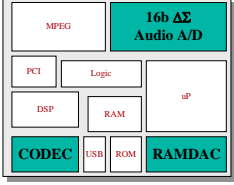
System-on-Chip Design Methodology


System Constructed from Virtual Components

- ◆ VCs have interfaces designed to standard busses for reuse
- ◆ Rapid assembly of system
- ◆ Architecture is often Software plus data converters

Circuit Implementation Tends to Evolve from TDD to SOC

- ◆ Processes improve
- ◆ Priority of performance decreases
- ◆ Priority of programmability increases



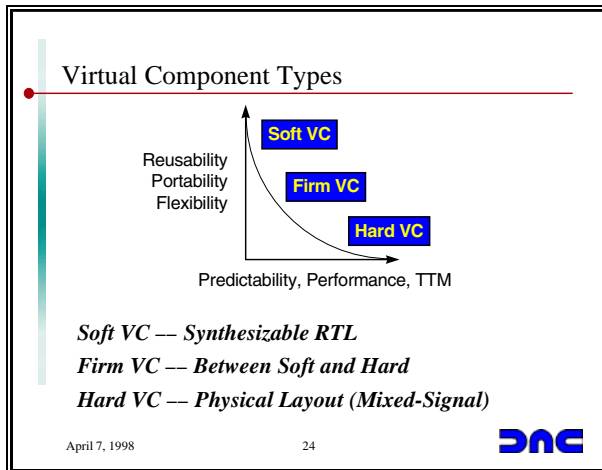
June 12, 1998
23


Slide 23

System-on-Chip Design Methodology

The system-on-chip design methodology differs from the block-based design methodology in that the blocks are virtual components, meaning that they are designed for reuse. This implies that they are compliant with VSI standards so that they can easily be wired together. The system-on-chip methodology is applied to very large systems that must be assembled very quickly. It provides faster time-to-market, but lower performance. It tends to be applied to large digital systems that have data converters at the interface to the outside world. As such, they tend to be highly programmable and flexible.

Modems today represent ideal candidates for the system-on-chip design methodology because they consist largely of digital circuitry (microcontroller, DSP, and bus interface) with an ADC and DAC at the interface to the phone line. Interestingly, modems used to be designed with a top-down methodology. However, as the technology improved, performance diminished as the highest priority and flexibility or re-programmability became more important. The signal processing changed from being largely analog to mainly digital. Thus, a block-based or system-on-chip design style now makes more sense than an MS top-down design style.

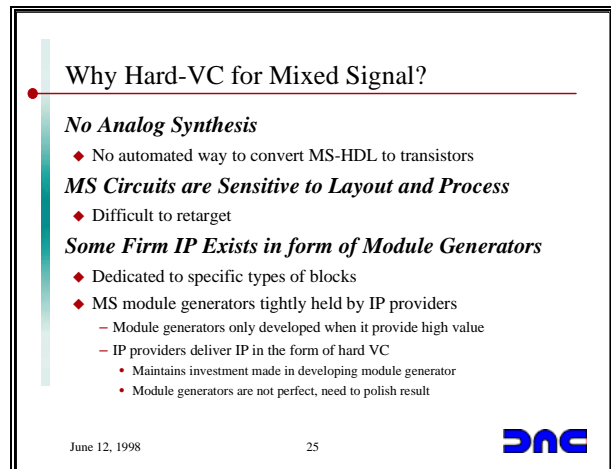


Slide 24

Virtual Component Types

Virtual components (VC) come in three basic types that are distinguished by what form they are delivered in. With hard VC, the actual layout of the block is fixed. Hard VCs can be digital or mixed-signal. With a soft VC, only synthesizable RTL is delivered. This is only for digital blocks. Firm VCs fit between soft and hard VCs. For example, a firm VC might consist of RTL plus floor planning information. Or it might consist of a parameterized module generator that automates the process of going from soft to hard.

Soft VCs are portable, easily reused, and flexible. By using different libraries, one can target different processes. It can also be modified to easily fit the target system. On the other hand, hard VCs are more predictable and provide better time-to-market and performance.



Slide 25

MS Virtual Components are Hard

In contrast to the digital world, there is no equivalent to a RTL representation for analog circuits, and there is no general-purpose analog synthesis. The nearest thing to synthesis in the analog world is a module generator that is targeted for particular type of block, such as switched-capacitor filters or data converters of a particular architecture.

Developing these module generators is expensive and difficult. The expense is only justified if the cost can be amortized over a large number of designs. In addition, the module generators are generally not good enough to deliver the final version of the block. Instead, the operator must generally polish the design to get it to meet the required performance specifications. Thus, generally module generators are expected to be developed and held tightly by IP providers, which use them as part of a design methodology that allows them to deliver IP very rapidly. The IP providers deliver their IP in the form of hard VC (layout) that is targeted to run on a specific process.

Importance of AMS Languages

Open Standard Languages: Verilog-AMS, VHDL-AMS

- ◆ Enables IP sharing
- ◆ Grow the market for mixed-signal simulators
 - Fund the development of single engine MS simulators

Common Language for Analog and Digital

- ◆ Supports a simple merged mixed-signal flow
- ◆ Natural for both analog and digital designers
- ◆ Natural support for mixed-signal blocks

AMS Languages Used for Verification

- ◆ Not for synthesis

April 17, 1998 26

Slide 26

MS-HDL Issues

Mixed-Level Simulation

- ◆ V*-AMS are behavioral languages
 - No built-in support for semiconductor models
 - Transistor models provided as 'built-ins' by simulator
 - Compatibility issues

Model Libraries for MS-SOC

- ◆ Each design is custom, and so each model must be hand crafted
- ◆ Most users write their own models
- ◆ Language must be very easy to learn and use

April 17, 1998 27

Slide 27

Importance of the AMS Languages

In 1998, Verilog-AMS is expected to be released by Open Verilog International (OVI) and VHDL-AMS is expected to be released by the IEEE as standard 1076.1. In 1999, several commercial implementations of the Verilog-AMS and VHDL-AMS standards are expected to become available. Collectively, Verilog-AMS and VHDL-AMS are referred to as V*-AMS or simply the AMS languages.

These languages are expected to have a big impact on MS-SOCs. Being open standards, they are expected to be widely supported. As such, they become useful for distributing models of IP, which allows easy evaluation and incorporation of IP into a SOC. In addition, more engineers are expected to learn and use these languages, making the market for MS simulators much larger. Expectations of this are causing simulator vendors to fund the development of single engine AMS simulators, which offer considerably improved performance and usability.

Having a common language for both analog and digital offers other key benefits. For example, it will be much easier to provide a single flow that naturally supports analog, digital and mixed-signal design. This makes it simpler for these designers to share their work. It also becomes substantially more straight-forward to write behavioral models for mixed-signal blocks.

However it is important to recognize that the AMS languages are primarily used for verification. Unlike the digital languages, the AMS languages will not be used for synthesis because the only synthesis that is available for analog circuits is very narrowly focussed.

MS-HDL Issues

Both Verilog-AMS and VHDL-AMS are mixed-signal behavioral languages. Other than allowing you to directly write the equations of your favorite MOS models, they provide no direct support for transistor-level simulation. Thus, a simulator that only supports an AMS language will not support mixed-level simulation, which is heavily used in mixed-signal top-down design (MS-TDD). However, most of the AMS simulators will be implemented with the popular semiconductor models built-in. The compatibility issues between simulation vendors involving semiconductor models that are common today will remain.

Because of rapid advances of semiconductor processes and the difficulty of re-targeting analog and mixed-signal designs, blocks intended for use in a MS-SOC tend to be much less heavily reused than those implemented as a packaged IC for use on a board. When reuse is high, as with packaged parts, it is feasible to dedicate a group of modeling experts to developing the models. However, when reuse is low, as in the MS-SOC case, this approach becomes too expensive because there are fewer uses over which to amortize the cost. Consider the MS-TDD design methodology where blocks are rarely reused. In this case, the designers themselves are typically expected to write their behavioral models. Thus, the language must be very easy to learn and use.

Verilog-AMS

Features

- ◆ Automatic interface element insertion
- ◆ Support for parasitic back-annotation
- ◆ Analog-only subset
- ◆ Easy to learn and use

June 12, 1998 28

Slide 28

Verilog-AMS

Verilog-A is an analog hardware description language patterned after Verilog-HDL. Verilog-AMS combines Verilog-HDL and Verilog-A into an MS-HDL that is a superset of both seed languages. Verilog-HDL provides event-driven modeling constructs, and Verilog-A provides continuous-time modeling constructs. By combining Verilog-HDL and Verilog-A it becomes possible to easily write efficient MS behavioral models. Verilog-AMS also automatic interface element insertion so that analog and digital models can be directly interconnected even if their terminal / port types do not match. It also provides support for back annotating interconnect parasitics.

models can be directly written in VHDL-AMS. Unlike with Verilog, there is no analog-only subset. This makes it more difficult to get initial simulators to the market, which is expected to slow adoption of VHDL-AMS.

VHDL-AMS inherits both the good and the bad aspects of VHDL. VHDL is strongly typed, which is a serious flaw for mixed-signal designs. You are not allowed to interconnect digital and analog ports, and there is no support for automatic interface element insertion. In fact, you are not even allowed to connect ports from an abstract analog model (a signal flow port) to a port from a low-level analog model (a conservative port). VHDL-AMS also does not provide support for back-annotate of RC interconnect. These represent fundamental flaws that will have to be overcome by a simulation environment, making VHDL-AMS much more dependent on its environment. This should further slow deployment of VHDL-AMS.

VHDL-AMS does inherent some nice features from VHDL, such as support for configurations and abstract data types.

VHDL-AMS

Features

- ◆ Support for configurations
- ◆ Abstract data types

April 17, 1998 29

Slide 29

VHDL-AMS

VHDL-AMS adds continuous time modeling constructs to the VHDL event-driven modeling language. Like Verilog-AMS, mixed-signal behavioral

MS-SOC: The Beginning of a Long Journey

Open Technical Issues

- ◆ Testing
- ◆ Bottom-Up Verification
- ◆ Coupling

The Value Proposition

- ◆ Improved productivity through reuse
- ◆ Reduced design time
- ◆ The ability to design much larger circuits

June 12, 1998 30

Slide 30

MS-SOC: The Beginning of a Long Journey

We are still early in the process of developing solutions to the MS-SOC problem. We certainly do not yet have all the answers. Nor is it likely that we know yet what all the problems will be. However, the march has begun and some progress has been made. That progress along with remaining challenges will be detailed by the remaining speakers. If we are successful, we will be rewarded with the ability to design complete mixed-signal system on chip faster

than we design much smaller and less complex mixed-signal chips today.