Principles of Top-Down Mixed-Signal Design

Ken Kundert
Designer's Guide Consulting, Inc.

Version 1, February 2003

With mixed-signal designs becoming more complex and pressure increasing to get the designs right on the first try, designers are finding that the traditional bottom-up design style is no longer adequate. They must adopt a more rigorous process for design and verification: top-down design. It involves more than simply a cursory design of the architecture before designing the blocks. Rather, it requires a design methodology that emphasizes communication and planning, and that smoothly transitions from initial concept to final implementation, with each step being verified with simulation.

This paper was presented in the short course on System-on-a-Chip Design *given at the International Solid-State Circuits Conference (ISSCC-2003). It derives from presentations that were entitled* A Formal Top-Down Design Process for Mixed-Signal Circuits *given at the* Advances in Analog Circuit Design *workshop (AACD-2000 in Germany) and at the* Electronic Design Processes *workshop (EDP-2001 in Monterey), and from a paper published in* Analog Circuit Design*, R.J. van de Plassche, J.H. Huijsing and W.M.C. Sansen (editors), Kluwer Academic Publishers, November 2000.*

Last updated on September 21, 2006. Contact the author via e-mail at ken@designers-guide.com.

1 Introduction

The semiconductor industry's growing ability to integrate functionality onto silicon requires that both the digital and analog circuits be increasingly integrated on the same chip. ASIC analyst Handal Jones predicts that by 2006 70% of ASICs will contain analog content, up from 17% in 1998.

Mixed signal plays, or will soon play, a critical role in every high-value market served by the electronics industry. In networking and wireless communications, it plays a central role. All communications systems must interface to the physical communications media, and those media are, by definition, analog. In addition, mixed-signal design is key to overcoming the communication bottlenecks that exist in all high-performance computing systems.

While mixed-signal circuitry is traditionally confined to the periphery of electronic systems, the amount of periphery is growing and invading the core. Previously, the mixed-signal content existed only at the periphery of the entire electronics system. Now, it is proliferating to the periphery of individual chips and will eventually propagate to the periphery of individual blocks within the chip itself. Increasing performance requirements for the overall system are requiring that the various parts of the circuit communicate more quickly. In the past, communication largely occurred over relatively slow externally-clocked buses. Now, design activity is focused on utilization of high-speed self-clocked serial links. For a given data transfer rate, the frequencies present in serial links are much higher than those found in busses and should reach 4-10 GHz within the next year or so. Because of the frequencies involved, and the fact that the signals travel over long non-ideal channels, each of these links involves a substantial amount of mixed-signal circuitry. As a result, virtually every chip within a computer will have a significant mixed-signal content. Unlike with many existing mixed-signal applications, such as wireless communications, in this case it is not possible to separate the mixed-signal circuitry onto a separate chip.

As an example of the trend toward the increasing use of mixed signal circuitry on previously all digital chips for high speed I/O, consider the new Stratix-GX high-end FPGA from Altera. It is a high-density programmable logic device, which are traditionally purely digital devices, but this one includes up to 45 1 Gb/s source synchronous I/O channels for chip-to-chip communications and up to 20 full duplex 3.125 Gb/s asynchronous transceiver channels for system-to-system communications.

1.1 Mixed-Signal Design

The mixed-signal design process has changed relatively little over the past two decades, and in comparison to the digital design process, is slow, labor intensive, and error prone. While digital designers have improved their design methodology and adopted design automation, analog and mixed-signal designers by and large have not.

There are two reasons why digital designers are far ahead of analog designers in improving their design processes. First, digital designers confronted the need to design very large and complex systems much earlier than analog designers. Consider that large digital chips today consist of tens of millions of transistors, while complex analog chips contain only tens of thousands of devices. Second, the digital design problem is much more amenable to automation than the analog problem.

Consider a digital design. In most cases digital systems are implemented as finite-state machines (FSM) and constructed from standard cell libraries. Using a FSM formulation acts to unify and homogenize digital design and gives it a well-understood mathematical foundation. This foundation was thoroughly explored in the late ’80s resulting in the commercial logic synthesis tools of the early ’90s. These tools take RTL, a relatively high-level description of a digital system that is created by designers and can be verified with the help of logic simulators, to produce an optimized gate-level description of the system. This transformation is possible because digital systems are constructed from a limited set of relatively simple and well-behaved building blocks. The building blocks of digital systems are gates and registers. The blocks, generally referred to as cells, all share a common I/O model and so are easily interconnected, are derived from a relatively small number of cell types that have very simple and easily described behavior, are easily parameterized in terms of the number of inputs and outputs, and have a simple and easily adjusted performance trade-off that involves only speed and power. Logic synthesizers operate by creating a complete mathematical description upon which it performs transformations in order to create an optimal design in terms of speed, power, and area. This is a two step process. First, equivalence transformations are applied to the mathematical descriptions in order to reduce the total number of gates, which minimizes the area, and the depth of the logic, which roughly maximizes the speed. This is possible because each block has a simple logical description and a common interface model. Then, the drive ability of each gate is adjusted to provide the lowest power while still meeting speed requirements. This is possible because this speed-power trade-off is easily made in each gate.

Now consider analog design. Analog design has no equivalent to finite-state machines, and so has no unified formulation and no common mathematical foundation. It also has no universal equivalence transformations that allow the topology of a circuit to be easily modified without risk of breaking the circuit. These problems prevent a topological mapping from a behavioral description to hardware. Even if one were mathematically possible, the lack of a common I/O model for analog blocks would prevent the topological modifications that are needed for either mapping or topology optimization.

It might be possible to try to enforce a common I/O model for analog circuits, but doing so would be very expensive. For example, one might simply specify that the signals at the inputs and outputs of analog blocks center around a particular value, have the same maximum swing, and that outputs have zero output impedance and inputs have zero input admittance. The problem is that doing so would necessitate extra circuitry in each analog block that is there simply to assure compliance to the I/O model. That circuitry reduces the overall performance of the circuit by increasing power dissipation, increasing noise, decreasing bandwidth, etc. This differs from the digital world where the common I/O model was achieved naturally and without significant cost. In addition, it is not possible to achieve these ideals at high frequencies. Instead, some common reference impedance would have to be specified, such the 50Ω used at the system level, but driving such loads greatly increases power dissipation.

Finally, there is no simple way to trade-off the various performance metrics that are important with analog blocks, which makes it very difficult to perform a parametric optimization. Sensitivity-based local optimizations can be used, but the improvement provided by these approaches is usually small. Monte Carlo-based global optimizers offer better improvements, but require substantially more in the way of computer resources.

The end result is that analog designers have no equivalent to RTL, a relatively high-level language in which they can describe their design and from which they can synthesize an implementation that is guaranteed to be functionally correct and have near optimal performance. As such they must transform their designs from concept to implementation by hand, and so the design process is naturally much slower and more error prone that the design process for digital circuits.

The outlook for providing the equivalent to logic synthesis for analog designers is bleak. However, things cannot continue as they are; the current situation is becoming untenable. While a complex digital chip can be designed correctly on the first try in a few months, designing a complex analog chip can require 3-4 respins and up to a year and a half to get right. This is problematic for many reasons:

1. The tremendous mismatch in schedule and risk between the analog and digital portions of a mixed-signal makes it difficult to justify combining analog and digital on the same chip.
2. The high risk makes planning difficult. It is hard to predict when product will be available, and when valuable analog designers will free up.
3. A long time-to-market makes it tough to react to changes in market trends and competitive pressures.
4. Analog and mixed-signal product development demands large investments of time and money. This makes it difficult to justify developing new analog products, especially in tough economic times.
5. Analog and mixed-signal designers are scarce and hard to recruit. Compounding this problem is the inherently low-level of productivity of the current mixed-signal design process, which makes it difficult for small design houses that are not focused on analog to field an analog design capability.
6. Some mixed-signal designs are becoming so large that, with the low productivity of the analog design process, a team of analog designers that is large enough to take on the project and complete it in a timely manner simply cannot be assembled.

Clearly a change is needed. It is interesting to note that when digital designers were trying to design systems of a size comparable to today's mixed-signal designs, their design process was not that different from what analog designers are using today. But it was at that point that they began to transition to a more structured and more automated design methodology. For analog designers, substantial automation may not be in the cards in the near future, but the need to transition to a more structured design methodology that is both more efficient and that allows designers to handle the growing size of analog and mixed-signal circuits is clearly needed.

The availability of logic synthesis tools was not the only enabling factor for digital designers to move to more efficient design methodologies. By moving to FSM and RTL, digital designers also gave up considerable performance in terms of speed and power. They made this sacrifice to be able to design the larger and more complex systems quickly. This sacrifice was a critically important enabling factor. Analog and mixed-signal designers have not demonstrated the willingness to make a similar sacrifice. In those cases where performance is not critical, the tendency is to instead convert the circuit to a digital implementation in order to gain flexibility. In the remaining cases sacrificing performance is not an option, however it is also not clear that such a sacrifice is needed. Analog designers do not have the equivalent of logic synthesis, so they will continue to use custom design methodologies. While moving to IP (intellectual property) reuse may

entail some sacrifice in overall system performance, changing to a top-down design methodology does not inherently imply lower system performance. In fact, the opposite is usually the case, using top-down design results in higher performance. Rather, the sacrifice that is demanded of analog and mixed-signal designers is that they must learn new skills, such as behavioral modeling, and they must be more disciplined in the way they design.

It is unlikely that analog and mixed-signal designers will ever be allowed on a large scale to trade any substantial amount of performance and power for a reduction in design time. Rather, in those cases where the performance and power requirements are not demanding, a digital implementation is usually preferred.

2 Design Drivers

Design of mixed-signal systems is getting more and more challenging, which is increasing the pressure to fix the analog design productivity problem. The challenges come in five different forms: the need to complete designs more quickly, the need to take on larger more complex designs, the need to increase the predictability of the design process, the need to reuse existing designs in order to more quickly and cheaply develop new designs, and the increasingly fluid nature of design and process requirements.

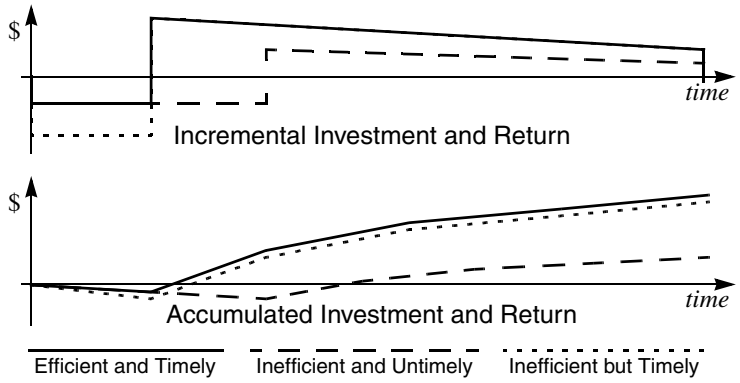
2.1 Getting to Market First

With the internet and wireless technology as the latest market drivers, the pace of the electronic marketplace continues to quicken. New products and new product categories are being created faster than ever before. In order to keep up with the rapid pace of the market, designers must get their products to market faster than ever. Those that are successful at bringing significant new capabilities to the market first are usually rewarded with higher profit margins and greater market share. Conversely, those that are late must face an uphill battle against entrenched competition. In fact, in markets where the product lifetime is short, the three month delay associated with a respin can be the difference between taking the market, and having no market at all.

To understand this, consider three scenarios for developing a product with Figure 1 showing the expected revenue for each scenario. For the first, consider employing an efficient product development process and being first to market. For the second, consider using the same number of developers with an inefficient development process, thereby causing the product to be late to market. This results in a much lower return because the product enters a market where a competitor has already established leadership position and because there are fewer available customers left. Finally, consider using an inefficient development process but increasing the number of developers in order to reach the market first. If this were possible, the development costs are higher, but the total return is almost the same as in the first case. This is because the returns are expected to be much greater than the initial development costs.

This example illustrates why it is often more important to get a product to the market first than it is to control development costs. Of course this assumes that the product is the right product in that it satisfies the customers needs, and that it has some new and valuable capability. With follow on products, the situation is somewhat different. Here,

FIGURE 1 *The expected investment and return for the same product developed using three different approaches.*



the market leadership position is largely determined and the need to develop the product in a timely manner is balanced by the need to control development costs.

To get a product to market in a timely manner one must have a design methodology that reduces the number of design and silicon iterations, maximizes the effectiveness of each designer, and makes it possible to effectively use more designers to speed the design by avoiding the "baby creation" problem. With the existing baby creation process, it takes nine months to create a new human baby. Adding more women to the process does not get the baby out any faster. To a large extent, the same is true with the current analog design process. Putting more designers on an analog design might get the design out somewhat sooner, but there are limits. Existing design methodology has limited opportunities for parallelism. In other words, there are several inherently serial tasks in the current design methodology that limit the reduction in time-to-market that results when adding more engineers to the product development team. A new design methodology is needed that is intrinsically more parallelizable.

2.2 Complexity

Moore's observation that the number of transistors available on an integrated circuit doubles every 18 to 24 months continues to hold. Competitive pressures compel designers to use these transistors to provide additional functionality and to increase the integration level and thereby decreasing the size, weight, power and cost of the product. As a result, designers are confronted with larger and more complex designs. The increasing size and complexity of these designs combines with the shrinking time available to develop and get them to market; making the job of the circuit designer today much more difficult than in the past.

Circuits are getting more complex in two different ways at the same time. First, circuits are becoming larger. Consider wireless products. 40 years ago a typical receiver contained between 5 and 10 transistors whereas it is common for a modern cell phone to contain 10 million transistors, an average growth rate of 30× per decade. Second, the operation of the circuits are becoming more complex. 30 years ago integrated circuits generally consisted of simple functional blocks such as op-amps and gates. Verification typically required simulating the block for two or three cycles. Today, mixed-signal chips implement complex algorithms that require designers to examine their operation

over several thousands of cycles, an average growth rate of 10× per decade. Examples include PLLs, ΣΔ converters, magnetic storage PRML channels, and CDMA transceivers. The result of these two effects together is that verification complexity is increasing at a blistering pace or roughly 300× per decade, and is outstripping the designers ability to keep up.

The CAD tools and computers employed by designers continually improve, which serves to increase the productivity of designers. However, the rate of productivity increase is not sufficient to allow the designers to keep up with the increasing complexity of designs and decreasing time-to-market requirements. The growing difference between the improvement in productivity needed to satisfy the demands of the market and the productivity available simply by using the latest CAD tools and computers is referred to as the design productivity gap. To close this gap, one must change the way design is done. As Dr. Henry Samueli, co-chairman and CTO of Broadcom, said at ISSCC '99, "Fundamental improvements in design methodology and CAD tools will be required to manage the overwhelming design and verification complexity" [26]. A design style that reduces the number of serial steps, increases the likelihood of first time working silicon, and increases the number of designers that can work together effectively is needed.

2.3 Avoidance of Risk

The recent downturn in the economy has exposed additional issues with the existing analog design process. With a tight economy, a long time-to-market is not the only reason why a good product might fail to exploit an available opportunity. It is also very important that the design make it to production with very few respins (and those respin should only involve changes to the metal mask), otherwise the opportunity might be lost due to a limited development budget. Budget limitations are exacerbated by the high cost of CMOS IC fabrication, and these costs are expected to grow rapidly, making respins increasingly problematic in the future. This is a particular problem for analog and mixed-signal design. For the same market value, mixed-signal design requires more resources, more talent, more design time, and hence, a larger development budget than digital design. The need for respins also injects substantial uncertainty into the planning process, which acts to increase the risk of designs that involve analog circuitry.

In uncertain economic times companies are much less likely to take on high-risk analog projects. Nor are they likely to add substantial risk to large digital designs by adding analog or mixed-signal circuitry. Rather, mixed-signal systems will be partitioned to the degree possible so that the analog circuitry is placed on a separate chip from the digital circuitry. Such a partitioning may be non-optimal in terms of power consumption and performance, but is deemed necessary to achieve acceptable levels of risk and cost. Once a working analog or mixed-signal chip is achieved, it will tend to be used repeatedly without change in successive versions of the system. Redesigning the analog portion of the system may result in lower system costs and improvements in performance, but simply takes too long and involves too much risk.

A more error prone design process also affects analog and mixed-signal design in another way. The errors that are common place in the early versions of mixed signal designs are often very difficult to track down. For this reason, mixed-signal designs often include additional circuitry that is used to help find these problems. This addi-

tional circuitry increases the area and, if used to monitor high speed nets, the extra loading can increase the power of a design and reduce its performance.

2.4 Reuse

An important piece of the strategy to increase the productivity of mixed-signal designers is reuse [6]. There are two types of reuse, IP (intellectual property) and derivatives. With IP, individual blocks are used in more than one system. With derivatives, an existing system is modified to fit the needs of a new application.

While IP reuse is promising, as yet the promise is largely unfulfilled. Part of the problem is the relentless pace of process technology evolution. For various reasons, it is currently difficult to migrate a mixed-signal designs from one process node to another, which limits the lifetime of IP. A limited lifetime also limits the degree one can leverage a particular piece of IP, which limits the marginal investment one can make to support reuse. Without an investment in documentation, modeling, and publishing, it is difficult for designers that are interested in incorporating existing blocks in their design to find potential IP and know whether it is suitable. The problem is heightened by the need share IP between companies, where competitive pressures force suppliers to make it easy to evaluate and integrate their IP.

Derivatives are more common, but currently require that the design team remains relatively intact, because much of the design knowledge that went into the system is never written down and it is difficult to pass on to a new team. This presents companies with an unpleasant choice. Generally the top design teams are the ones to take on the large new systems that will lead to several derivatives. Once the initial design is complete, should the design team continue to work on derivatives, in which case they would not be available to take on the design of new systems. Or do they move on to the new systems, in which case the derivatives will at best take considerably longer and may flounder.

2.5 Increasingly Fluid Design and Process Requirements

Between time-to-market pressures and the need to support reuse, designers are increasingly being faced with having to design systems with either the design requirements or the process shifting during the design process, or with the requirement that the design be easily migrated to a new application or process after completion of the initial version.

As mentioned before, mixed-signal design is increasingly focussed on communications, and communications design is dominated by industry standards. Time-to-market is particularly challenging when trying to design to standards, because the design process usually must begin before the standard is finalized, meaning that the design requirements often change late in the design process. The design team must be able to quickly react to those changes, and still produce a high quality design without the errors that would cause a respin. Thus, a design methodology is needed that is tolerant to late changes, and that allows a design to be quickly and completely reverified after such changes.

As CMOS processes shrink, secondary effects (short channel effects, mobility reduction, coupling, etc.) become increasingly important. It is difficult to anticipate the effect of these on the design at the beginning of the design process. The design components need to be repeatedly simulated with estimated or extracted layout parasitics, and iteratively refined. As a consequence, the system design often needs to be revisited. While

this technically is not a change in the requirements or the process, it has a similar effect on the design process. Rather than it being a change in the process, it is a change in the understanding of how the process affects the design, and the design team must react to this change midway through the design process.

2.6 Gigabit I/O

In many applications, one can make an economic judgement as to whether it is better to put the mixed-signal circuitry on the same chip as the rest of the system to reduce costs, or to put it on a separate chip to reduce risk. This is the case, for example, with most wireless systems. However, with high speed serial I/O there is no choice available, the mixed-signal circuitry must go on the same chip. Thus, the large system-chip inherits the risk associated with mixed-signal circuitry. Given the large opportunities and investments[1], and the short market windows generally associated with these chips, this is not acceptable. As mentioned before, a respin can be the difference between capturing a market and having no market at all.

3 Traditional Approaches to Mixed-Signal Design

At the Design Automation Conference in 1998, Ron Collett of Collett International presented findings from a 1997 productivity study in which his firm analyzed 21 chip designs from 14 leading semiconductor firms. The study revealed a productivity gap of 14× between the most and least productive design teams. The study also revealed that developing analog and mixed-signal circuitry requires three to seven times more effort per transistor than designing digital control logic, though this factor was normalized out of the 14× ratio.

The reason for the poor productivity of those at the bottom of the scale are increasingly complex designs combined with a continued preference for bottom-up design methodology and the occurrence of verification late in the design cycle, which leads to errors and respins. There's a huge disparity in productivity between those mixed-signal designers who have transitioned to an effective “top-down” design methodology, and those who practice “bottom-up” design and rely solely on SPICE.

3.1 Bottom-Up Design

The traditional approach to design is referred to as bottom-up design. In it, the design process starts with the design of the individual blocks, which are then combined to form the system. The design of the blocks starts with a set of specifications and ends with a transistor level implementation. Each block is verified as a stand-alone unit against specifications and not in the context of the overall system. Once verified individually, the blocks are then combined and verified together, but at this point the entire system is represented at the transistor level.

1. A 0.25µ mask set costs $150K and includes 15 masks. A 0.18µ mask set costs $250K and includes 21 masks. A 0.13µ mask set costs $600K. At 90nm, masks are expected to cost $1.5M and at 65nm they will cost $4M [9].

While the bottom-up design style continues to be effective for small designs, large designs expose several important problems in this approach.

1. Once the blocks are combined, simulation takes a long time and verification becomes difficult and perhaps impossible. The amount of verification must be reduced to meet time and compute constraints. Inadequate verification may cause projects to be delayed because of the need for extra silicon prototypes.
2. For complex designs, the greatest impact on the performance, cost and functionality is typically found at the architectural level. With a bottom-up design style, little if any architectural exploration is performed, and so these types of improvements are often missed.
3. Any errors or problems found when assembling the system are expensive to fix because they involve redesign of the blocks.
4. Communication between designers is critical, yet an informal and error prone approach to communication is employed. In order to assure the whole design works properly when the blocks are combined, the designers must be in close proximity and must communicate often. With the limited ability to verify the system, any failure in communication could result in the need for additional silicon prototypes.
5. Several important and expensive steps in the bottom-up design process must be performed serially, which stretches the time required to complete the design. Examples include system-level verification and test development.

The number of designers than can be used effectively in a bottom-up design process is limited by the need for intensive communication between the designers and the inherently serial nature of several of the steps. The communication requirements also tend to require that designers be co-located.

3.2 Moving to Top-Down Design

In order to address these challenges, many design teams are either looking to, or else have already implemented, a top-down design methodology [1,5,7,19,21]. In a primitive top-down approach [4], the architecture of the chip is defined as a block diagram and simulated and optimized using a system simulator such as Matlab or Simulink. From the high-level simulation, requirements for the individual circuit blocks are derived. Circuits are then designed individually to meet these specifications. Finally, the entire chip is laid out and verified against the original requirements.

This represents the widely held view of what top-down design is. And while this is a step towards top-down design, it only addresses one of the issues with bottom-up design (point 2 in Section 3.1). In essence, these design groups have not fundamentally changed their design process, they have simply added an architectural exploration step to the front. The flaw in this approach is that there is a important discontinuity in the design flow that results because the representation used during the architectural exploration phase is incompatible with the representation used during implementation. This discontinuity creates two serious problems. First, it leaves the block designers without an efficient way of assuring that the blocks all work together as expected. One could assemble transistor-level representations of the blocks and run simulations, but the simulations are too slow to be effective. The first time the blocks can be thoroughly tested together is first silicon, and at that point any errors found trigger a respin. Second, the discontinuity makes communications more difficult and ad hoc and so acts to separate

the system designers from the circuit designers, and the circuit designers from each other. Without a reliable communication channel, designers resort to using verbal or written specifications, which are often incomplete, poorly communicated, and forgotten half way through the project. It is the poor communication process that creates many of the errors that force respins, and the separation that allows the errors to hide until the design is available as silicon.

To overcome these issues, one needs a design methodology that

1. Improves communications between designers (between system and block designers, between block designers, between current designers and future designers (to support reuse).
2. Eliminates the discontinuity that acts to hide errors and separate the system designers from the block designers.
3. Improves verification so that it finds the errors that cause respins, and finds them earlier so that they are less disruptive and easier to fix.
4. Improve designer effectiveness.
5. Reorganize the design tasks, making them more parallel and eliminating long serial dependencies.
6. Reduce the need for extensive transistor-level final verification.
7. Eliminate respins!

RF designers typically use this type of primitive top-down design approach. They begin with the system design. Typically using a spreadsheet, the gain, noise figure and distortion budget is explored, and with the help of guides like the Friis equation, is shared amongst the various blocks of the receiver. The design is then iterated until the performance of the system as predicted by the spreadsheet is met and the performance requirements on the blocks are reasonable. At this point, the design proceeds bottom-up relying solely on transistor-level design. Eventually, the spreadsheet is updated with the actual values coming from the transistor-level simulation, and if the system performance is not satisfactory the process repeats. The problem is that even when using the updated results, the performance predicted by the spreadsheet will not match the results achieved in silicon. This happens because of miscommunications, either in the meaning or the actual values of the block specification, and because the system-level description is crude and does not account for things like loading effects. When designing non-integrated receivers this is not as problematic because all the stages are generally designed for power matching and the voltage supply is reasonably high ($V_{dd} \geq 5$ V). In CMOS design the voltage supply is low (1.2 V in a 0.13 μm process) and the blocks do not share matched impedances. The result, of course, is that multiple silicon iterations are needed to achieve the required system performance levels.

4 Principles of Top-Down Design

A well designed top-down design process methodically proceeds from architecture- to transistor-level design. Each level is fully designed before proceeding to the next and each level is fully leveraged in design of the next. Doing so acts to partition the design into smaller, well defined blocks, and so allows more designers to work together productively. This tends to reduce the total time required to complete the design. A top-down design process also formalizes and improves communications between designers.

This reduces the number of flaws that creep into a design because of miscommunication. The formal nature of the communication also allows designers to be located at different sites and still be effective.

Following a top-down design methodology also reduces the impact of changes that come late in the design cycle. If, for whatever reason, the circuit needs to be partially redesigned, the infrastructure put in place as part of the methodology allows the change to be made quickly. The models can be updated and impact on the rest of system can be quickly evaluated. The simulation plan and the infrastructure for mixed-level simulations is already be available and can be quickly applied to verify any changes.

An effective top-down design process follows a set of basic principles.

1. A shared design representation is used for the entire length of the project that allows the design be simulated by all members of the design team and in which all types of descriptions (behavioral, circuit, layout) can be co-simulated.
2. During the design process each change to the design is verified in the context of the entire, previously verified, design as dictated by the verification plan.
3. A design process that includes careful verification planning where risks are identified up-front and simulation and modeling plans are developed that act to mitigate the risks.
4. A design process that involves multiple passes, starting with high level abstractions and refining as the detail becomes available. In effect, running through the entire process very quickly at the beginning with rough estimates and guesses to get a better understanding and better estimates, and then refining the design as the process progresses.
5. To the degree possible, specifications and plans should be manifested as executable models and scripts, things that are used in the design process on a daily basis, rather than as written documents.

4.1 A Shared Design Representation

In the primitive top-down design process commonly used today, the system designers use a different design representation than the circuit designers. For example, the system designers might use a spreadsheet, Matlab, Simulink, SPW, or SystemView while the circuit designers would use Verilog, VHDL, or SPICE. This causes a myriad of problems, perhaps the most important being that they are using different tools to explore the design that make it difficult for them to share what they learn during the design process. As mentioned before, this leads to communication problems and eventually to design errors that are generally not caught until first silicon.

If instead a common simulatable design representation is used, such as Verilog-AMS or VHDL-AMS, then the system engineers can build an architectural-level description of the design constructed from behavioral models of each of the blocks that can be evaluated by each of the circuit designers. In effect, the circuit designers start by receiving an executable example of what they are expected to design. If they have trouble meeting their assigned specifications, they can go back to the system engineers with simulations that show how the system is affected by the shortfall. Since both types of engineers are working in a familiar environment, communication is enhanced and potential resolutions can be explored together. The ready availability of behavioral models of the blocks that act as executable examples greatly reduces the need for onerous specifications that

describe the desired behavior of each block, specifications that were often poorly written and that frequently go unread.

4.2 Every Change is Verified

In a primitive top-down design methodology, the architectural description of the system is usually thoroughly verified using simulation. However, the design is then re-created at the circuit level during the implementation phase and this version of the design is never checked against the original architectural description. This discontinuity is where many of the errors creep in that are not found until first silicon. In effect, the verification that was done in the architectural phase is not leveraged during the implementation phase, and verification in the implementation phase is generally not as effective because it is very slow and so cannot be as comprehensive. In addition, the test benches used during the architectural design phase often cannot be reused during the implementation phase, and are generally difficult to re-create.

It is important instead to use a common simulatable representation for the design that allows both the system-level and circuit-level descriptions of the various blocks to be co-simulated, as is possible with languages such as Verilog-AMS or VHDL-AMS. This capability is referred to as mixed-level simulation [15,24]. With it, individual blocks or small sets of individual blocks can be represented at the transistor- or even layout-level and be co-simulated with the rest of the system, which is described with high-level behavioral models. While these simulations are often considerably slower than simulations where every block is described at the high-level, they are also considerably faster than simulations where every block is described at the transistor level. And they allow the individual blocks to be verified in a known-good representation of the entire system. In effect, the system simulations are leveraged to provide an extensively verified test bench for the individual blocks.

Consider a simple example. It is not uncommon for a system to fail at first silicon because of a miscommunication over the polarity of a digital signal, such as a clock, enable, or reset line. Such errors cannot survive in the high-level description of the system because of the extensive testing that occurs at this level. They also cannot survive during mixed-level simulation because the individual block, where the error is presumed to be, is co-simulated with shared models for which the polarity of the signal has already been verified. They can, however, survive in either a bottom-up or primitive top-down design process because the test benches for the individual blocks are created by the corresponding block designers. Any misunderstanding of the required interface for the block will be reflected both in the implementation of the block and in its test bench, and so will not be caught until first silicon.

4.3 Verification Planning

Generally users of bottom-up or primitive top-down design methodologies find that the errors detected at first silicon are a result of rather mundane mistakes that occur at the interfaces between the various blocks. These errors are generally caused by communication breakdowns and would have been easy to find with simulation had anyone thought to look for them. The fact is that the focus of verification efforts in these methodologies is on guaranteeing the performance of the individual blocks, and not on identifying the problems that result when the blocks are interconnected. Some effort is generally spent on trying to verify the system as a whole, but it comes late in the process when the sys-

tem is represented largely at the transistor level. At this stage, the simulations are quite slow and the amount of functionality that can be verified is very limited.

In a well-conceived top-down design process a verification planning step occurs that focuses on anticipating and preventing the problems that occur when assembling the blocks into a system. In order to be effective, it must move the verification to as early in the design process as possible and occur with as much of the system described at a high level as possible. Moving the chip-level verification up in the design process means that errors are caught sooner, and so are easier and less expensive to fix. Using high-level models means that the simulations run faster, and so can be substantially more comprehensive.

In a zealousness to accelerate the simulation, care must be taken to assure that enough of the system is at the right level to assure that the desired verification is actually occurring. Thus, the verification plans must include both simulation plans, that describe how the verification is to occur, and modeling plans, that indicate which models need to available to support the verification plan and which effects should be included in the models. The modeling plan is very important. Without it behavioral models may be written that do not include the desired effect while including many effects that are unrelated to what is being verified. If they do not model the desired effect, then the verification will not be effective, if they model too many effects, then the verification runs unnecessarily slow and the models become more difficult and expensive to develop. The goal with the modeling plan is to identify a collection of simple models along with directions as to when they should be used, rather that trying to develop one complex model that is used in all cases.

An important benefit of the verification plan is that it allows the design team to react to late changes in the design requirements with confidence. When a change to the requirements occur, it is possible to quickly revisit the verification plan, modify the design, update the models, and then apply it to the amended design to assure it satisfies the new requirements. Since it spells out all the simulations that need to occur to verify the design, there is little chance that a change needed by the new requirements that happens to break some other part of the design will go unnoticed.

Another important benefit of the up-front planning process used when developing the verification plan is that it tends to sensitize the design team to possible problem areas, with the result that those areas are less likely to become problems.

4.4 Multiple Passes

To reduce the risk of design iterations that result from unanticipated problems, it is important to take steps to expose potential problems early by working completely through an abstract representation of the design, using estimates as needed. As the design progresses and more detailed and reliable information becomes available, the abstract representation is successively refined. This process begins by developing a top-level behavioral model of the system, which is refined until it is believed to be an accurate estimate of the desired architecture. At this point, there should be reasonable understanding as to how the blocks will be implemented, allowing size estimates to be made for the blocks, which leads to an initial floorplan. Top-level routing is then possible, which leads to parasitic extraction, with the effect of the parasitics being back annotated to the top-level. Simulations can then expose potential performance problems as a result of the layout, before the blocks are available. This may result in early changes to the

architecture, changes in block specifications, or perhaps just an improvement of the verification plan. However, these changes occur early in the design process, which greatly reduces the amount of redesign needed.

As the blocks are implemented and more information becomes available, the process is repeated if there are any surprises.

4.5 Executable Specifications and Plans

When a design fails because of miscommunications between engineers, it is a natural reaction to insist that in future designs formal specifications and plans be written in advance as a way of avoiding such problems. In practice, this does not work as well as generally hoped. The act of writing things down is beneficial as it gets the engineers thinking more deeply about their designs up-front, and so they develop a better understanding of what is expected and what could go wrong. However, as for the written specifications and plans themselves, they can take a long time to write, are usually not very well written or maintained, and are often not read by the other engineers. The fact is, the documents themselves are rarely effective at improving the communications between the engineers, rather it is the better understanding that comes from writing them that acts to improve communications.

If instead, specifications and plans took the form of executable models and scripts that would be used and valued on a daily basis, perhaps with a small amount of accompanying documentation, then they would be naturally well written, well used, and well maintained. The models and scripts are also inherently very specific, which eliminates the ambiguity that occurs in written documents and that can result in misunderstandings that lead to respins. These models and scripts should be maintained with the design data and shared between all designers. This avoids another of the problem with written specifications; the situation where one engineer is unaware that another has updated a specification.

Use of executable specifications and plans in the form of models and scripts both substantially improves the design process for the initial version of the chip, as well as greatly easing reuse of either the design as a whole, or the blocks used in constructing the chip. IP reuse, or reuse of the blocks, it made considerably easier because validated high-level models of the blocks are available at the end of the design process. These models would then be used to easily evaluate the blocks as to there suitability for use in other designs. Derivatives, or system reuse, is greatly simplified by the existence of all of the models and scripts. It makes it much easier for either a new team, or new team members, to get an quick understanding of an existing design and initiate the process of making changes to retarget the design to a new application. Furthermore, having models and verification scripts that have been refined by the experiences of the first design team make it more likely that the follow-on designs will debut without surprises.

5 A Rigorous Top-Down Design Process

The rigorous top-down design methodology described here is a substantial refinement of the primitive top-down process described in Section 3.2. It follows the principles described in Section 4 in order to address all of the problems associated with bottom-up design, as identified in Section 3.1.

5.1 Simulation and Modeling Plans

An important focus in a good top-down design methodology is the development of a comprehensive verification plan, which in turn leads to the simulation and modeling plans. The process begins by identifying particular areas of concern in the design. Plans are then developed for how each area of concern will be verified. The plans specify how the tests are preformed, and which blocks are at the transistor level during the test. For example, if an area of concern is the loading of one block on another, the plan might specify that one test should include both blocks represented at the transistor level together. For those blocks for which models are used, the effects required to be included in the model are identified for each test. This is the beginning the modeling plan. Typically, many different models will be created for each block.

It is important to resist the temptation to specify and write models that are more complicated than necessary. Start with simple models and only model additional effects as needed (and as spelled out in the modeling plan). Also, the emphasis when writing models should be to model the behavior of the block, not its structure. A simple equation that relates the signals on the terminals is preferred to a more complicated model that tries to mimic the internal working of the block. This is counter to the inclination of most designers, whose intimate knowledge of the internal operation of the block usually causes them to write models that are faithful to the architecture of the block, but more complicated than necessary.

It is also not necessary to model the behavior of a circuit block outside its normal operating range. Instead, you can add code in a model that looks for inappropriate situations and reports them. Consider a block that supports only a limited input bias range. It is not necessary to model the behavior of the block when the input bias is outside the desired range if in a properly designed circuit it will never operate in that region. It is sufficient to simply generate a warning that an undesirable situation has occurred.

Following these general rules will result in faster simulations and less time spent writing models. However, the question of how much detail is needed in each model is a delicate one that must be answered with great care. It is important to understand the imperfection in the blocks and how those imperfections affect the overall performance of the system before one can know whether the effects should be included in a model. Also, it is not always true that a pure behavioral model will be superior to a more structurally accurate model. Often making the model more structurally accurate makes it more predictive, and also may make it easier to include some secondary effects due to parasitics.

The simulation plan is applied initially to the high-level description of the system, where it can be quickly debugged. Once validated, it can then be applied to transistor level simulations.

A formal planning process generally results in more efficient and more comprehensive verification, meaning that more flaws are caught early and so there are fewer design iterations.

5.2 System-Level Verification

System-level design is generally performed by system engineers. Their goal is to find an algorithm and architecture that implements the required functionality while providing adequate performance at minimum cost. They typically use system-level simulators, such as Simulink [22] or SPW [3], that allow them to explore various algorithms and

evaluate trade-offs early in the design process. These tools are preferred because they represent the design as a block diagram, they run quickly, and they have large libraries of predefined blocks for common application areas.

This phase of the design provides a greater understanding of system early in the design process [13,14]. It also allows a rapid optimization of the algorithm and moves trades to the front of design process where changes are inexpensive and easy to make. Unworkable approaches are discarded early. Simulation is also moved further up in the design process where it is much faster and can also be used to help partition the system into blocks and budget their performance requirements.

Once the algorithm is chosen, it must be mapped to a particular architecture. Thus, it must be refined to the point where the blocks used at the system level accurately reflect the way the circuit is partitioned for implementation. The blocks must represent sections of the circuit that are to be designed and verified as a unit. Furthermore, the interfaces must be chosen carefully to avoid interaction between the blocks that are hard to predict and model, such as loading or coupling. The primary goal at this phase is the accurate modeling of the blocks and their interfaces. This contrasts with the goal during algorithm design, which is to quickly predict the output behavior of the entire circuit with little concern about matching the architectural structure of the chip as implemented. As such, mixed-signal hardware description languages (MS-HDLs) such as Verilog-AMS [20,25] or VHDL-AMS [18] become preferred during this phase of the design because they allow accurate modeling of the interfaces and support mixed-level simulation.

The transition between algorithm and architecture design currently represents a discontinuity in the design flow. The tools used during algorithm design are different from the ones used during architecture design, and they generally operate off of different design representations. Thus, the design must be re-entered, which is a source of inefficiencies and errors. It also prevents the test benches and constraints used during the algorithm design phase from being used during the rest of the design.

On the digital side, tools such as SPW do provide paths to implementation via Verilog and VHDL generation. Similar capabilities do not yet exist for the analog or mixed-signal portions of the design. An alternative is to use Verilog-AMS or VHDL-AMS for both algorithm and architecture design. This has not been done to date because simulators that support these languages are still relatively new. It will probably take a while for this approach to become established because of the absence of application specific libraries needed for rapid system-level exploration. Alternatively, a simulator like AMS Designer from Cadence that supports both algorithm and architecture development by combining SPW with Verilog-AMS can be used [2].

5.3 Mixed-Level Simulation

Without analog synthesis, analog design is done the old fashioned way, with designers manually converting specifications to circuits. While this allows for more creativity and gives higher performance, it also results in more errors, particularly those that stem from miscommunication. These miscommunications result in errors that prevent the system from operating properly when the blocks are assembled even though the blocks were thought to be correct when tested individually.

To overcome this problem, mixed-level simulation is employed in a top-down design methodology for analog and mixed-signal circuits. This represents a significant but

essential departure from the digital design methodology. Mixed-level simulation is required to establish that the blocks function as designed in the overall system.

To verify a block with mixed-level simulation, the model of the block in the top-level schematic is replaced with the transistor level schematic of the block before running the simulation. For this reason, all of the blocks in the architectural description of the system must be "pin-accurate", meaning that they must have the right number of pins and characteristics of each pin must be representative of the expected signal levels, polarities, impedances, etc.

The pin-accurate system description, 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 requires that both the system and the block designers use the same simulator and that it be well suited for both system- and transistor-level simulation.

Mixed-level simulation allows a natural sharing of information between the system and block designers. When the system-level model is passed to the block designer, the behavioral model of a block becomes an executable specification and the description of the system becomes an executable test bench for the block. When the transistor level design of the block is complete, it is easily included in the system-level simulation.

Mixed-level simulation is the only feasible approach currently available for verifying large complex mixed-signal systems. Some propose to use either timing simulators (sometimes referred to as fast or reduced accuracy circuit simulators) or circuit simulators running on parallel processors. However, both approaches defer system-level verification until the whole system is available at transistor level, and neither provide the performance nor the generality needed to thoroughly verify most mixed-signal systems. They do, however, have roles to play both within the mixed-level simulation process and during final verification.

Successful use of mixed-level simulation requires careful planning and forethought (provided during the verification planning process). And even then, there is no guarantee that it will find all the problems with a design. However, it will find many problems, and it will find them much earlier in the design process, before full-chip simulations, when they are much less costly to fix. And with mixed-level simulation, it is possible to run tests that are much too expensive to run with full-chip simulation.

5.3.1 Mixed-Level Simulation Example

Though this example is several years old, it is representative of the type of circuit complexity that are mainstream today. It is a PRML channel chip that is difficult to simulate for two reasons. First, it is a relatively large circuit that involves both analog and digital sections that are closely coupled. Second, the architecture involves complex feedback loops and adaptive circuits that take many cycles to settle. The combination of many transistors and many cycles combines with the result being a simulation that is so expensive as to be impractical. In this case, the expected simulation time was predicted to be greater than a month.

The traditional approach to simulating a complex circuit like this is to simulate the blocks individually. Of course this verifies that the blocks work individually, but not together. In addition, for this circuit it is difficult to verify the blocks when operating

outside the system, and it is difficult to predict the performance of the system just knowing the performance of the individual blocks.

When the architecture was simulated at a high level with each block represented by a pin-accurate behavioral model, the simulation time was less than 10 minutes. Then, when a single block was run at the transistor level, the simulation ran overnight. Even though the full system was never simulated at the transistor level, it worked the first time because this methodology verified the blocks in the context of the system and it verified the interfaces between the blocks.

5.4 Bottom-Up Verification

Once a block is implemented, one could update the models that represent it to more closely mimic its actual behavior. This improves the effectiveness of mixed-level and system-level simulation and is referred to as bottom-up verification. To reduce the chance of errors, it is best done during the mixed-level simulation procedure. In this way, the verification of a block by mixed-level simulation becomes a three step process. First the proposed block functionality is verified by including an idealized model of the block in system-level simulations. Then, the functionality of the block as implemented is verified by replacing the idealized model with the netlist of the block. This also allows the effect of the block's imperfections on the system performance to be observed. Finally, the netlist of the block is replaced by an extracted model. By comparing the results achieved from simulations that involved the netlist and extracted models, the functionality and accuracy of the extracted model can be verified. From then on, mixed-level simulations of other blocks are made more representative by using the extracted model of the block just verified rather than the idealized model.

Bottom-up verification should not delayed until the end of the design process, but should rather be done continuously during the entire design process. Once a block has been implemented to the degree that a more representative model can be extracted, that model should replace the idealized top-level model as long as it does not evaluate substantially slower. Doing so tends to improve the effectiveness of mixed-level simulation and the accuracy of the extracted models. Thus, as a side benefit, the models that would be needed if the block were to be made into a shared IP block are already available and tested at the end of the project. If the model development for bottom-up verification were postponed to the end of the design process, the natural pressure to meet schedule targets as designs near tape-out often result in some of the verification, and perhaps all of the modeling, being skipped. This increases the chance of error and decreases the opportunity for reuse.

When done properly, bottom-up verification 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 and reuse.

5.5 Final Verification

In a top-down design process, SPICE-level simulation is used judiciously in order to get its benefits without incurring its costs. All blocks are simulated at the transistor level in the context of the system (mixed-level simulation) in order to verify their functionality

and interface. Areas of special concern, such as critical paths, are identified up front in the verification plan and simulated at the transistor level. The performance of the circuit is verified by simulating just the signal path or key pieces of it at the transistor level. Finally, if start-up behavior is a concern, it is also simulated at the transistor level. The idea is not to eliminate SPICE simulation, but to reduce the time spent in SPICE simulation while increasing the effectiveness of simulation in general by careful planning.

It is in this phase that the dynamic timing simulators (fast reduced-accuracy transistor-level simulators) play an important role. They often have the capacity to simulate large mixed-signal systems at the transistor level for a reasonable period of time. Again, even with timing simulators the simulations are generally only fast enough to provide limited verification. So use of a timing simulator does not offset the need for mixed-level simulation.

5.6 Test

During the design phase, the test engineers should use top-level description of the design as a simulatable prototype upon which to develop the test plan and test programs. The availability of a working model of the system early in the design process allows test engineers to begin the development and testing of test programs early. Moving this activity, which used to occur exclusively after the design was complete, so that it starts at the same time the block design begins significantly reduces the time-to-production [10,11,27]. Bringing test development into the design phase can reduce post-silicon debug time by 50% and can eliminate a turn by finding chips that are untestable early. It can also improve tests, which then improves yield.

6 Roles and Responsibilities

Current systems-on-chip (SoC) designs are in general very complex with the mixed-signal portion just a piece of a much larger whole. Within the mixed-signal design team, there are a number of individuals fulfilling leadership roles and responsibilities that are necessary to assure the success of the entire SoC. Those roles and responsibilities are attributed to individuals and described next. However, this is just one way they could be divided up between the team members. It is also not necessarily the best way, which would depend on the particular strengths of the team members as well as the type of design and size of the design team.

6.1 Team Lead

The team lead would supervise one, and only, one design project at a time. He or she works with the program manager to manage the schedule and the resources, acts as the technical interface to the customer, and is ultimately responsible for the technical execution of the implementation. As the primary driver of the implementation, the team lead is responsible for the developing the simulation and modeling plans, including the process corners, and managing the relationships and communications between the block designers. He or she also interfaces with the design methodology group and is responsible for keeping the process libraries and design kits updated. Finally, the team lead works closely with the system designer, the top-level designer, and the architect to specify the overall architecture, to negotiate the overall specifications as well as the specifi-

cations on the individual blocks, and to develop the test plan. In addition, the team lead would work with the top-level designer on the power supply, bias and clock distribution.

6.2 Top-Level Designer

The top-level designer works on a single project at a time and is responsible for specifying how the various blocks that make up a design will be assembled. As such, he or she owns the floorplan, and the inter-block wiring and helps the system designer with the architecture. The top-level designer is responsible for the top-level verification and mixed-level simulation, and so owns the responsibility for developing the overall test bench. He or she also designs the power supply, bias and clock distribution network with help from the team lead, and optionally works on the overall architecture, specifications, and test plan with the team lead, the system designer, and the architect.

The top-level designer owns the top-level schematic for the design. This schematic must be captured before any block design begins, even though it is likely to change before the design is complete. The top-level schematic specifies the partitioning of the design into blocks and the interface for each block. So each block should be “pin-accurate”. By this is it meant that in the top-level schematic, each block, and each pin on each block, is represented, and the type of each pin is carefully defined and documented. For example, an enable line on a block may be denoted “3V CMOS active high” or a trigger line may be described with “5V TTL positive edge triggered”. In this way, the top-level schematic provides clarity of intention to the design team. As the owner of the top-level schematic, the top-level designer must approve any changes to the interfaces of the blocks, and when changes occur must coordinate the updating and distribution of the new top-level schematic and models to the design team.

Most top-level simulation, especially in the final phase of the project, involves a lot of transistor-level simulations with extracted parasitics. For this reason the top-level designer is in general a different individual from the system designer. System designers tend to be more knowledgeable of architectures and algorithms. The top-level designer is in general an experienced block level designer that understands physical design very well, but he also has knowledge of the architecture. Top-level simulations can be very tricky. It is important that the top-level designer be facile with circuit simulators in order to overcome simulation problems and to obtain result in a reasonable time and with enough accuracy. These do not need to be run for every process corner but only for the one or two that are the most critical.

6.3 System Designer

The system designer tends to work on multiple projects simultaneously and is responsible for the algorithm development, the architecture, and system-level simulation (using tools such as SPW, Matlab, Simulink, SystemView, Excel, Verilog, Verilog-AMS, software written ad-hoc, etc.). He or she also optionally works on partitioning of performance specifications to the various blocks and the test plan with the team lead, the top-level designer, and the architect.

6.4 Block Designers

The block designers tend to work on a single project at a time and are responsible for one or more block designs within that project. As such, they take care of all aspects of

implementing the individual blocks, such as circuit design and layout. They are responsible for assuring that their blocks meet the required specifications, and that they work properly when embedded in the overall system. As such, they are expected to verify their designs in the context of the whole system using mixed-level simulation. Block designers also often take responsibility for building the behavioral models of their blocks that are shared with the rest of the design team.

6.5 Modeling Engineers

Modeling engineers may be, but are not usually, dedicated to individual projects. They are often used to develop the sophisticated models that would be difficult for the other members of the team. They are made available to the design teams with the recognition that modeling is a skill and discipline that is distinct from design, and that most design engineers do not have the knowledge or experience to create complex or difficult models.

6.6 Test Engineers

Test engineers work with the team lead and system and top-level designers and occasionally with the block designers to develop a test plan, and to specify self-test strategies, and to write the test programs. They may be shared between multiple projects, but should become involved at the onset of the project in order to assure that the chip is as testable as possible, and to reduce the time to market. For example, once the system-level model is available, the development of the test programs can begin. In this way, a formerly serial task that occurred after silicon became available can be done in parallel with the block design and assembly.

6.7 Senior Architect

The senior architect acts as a consultant to multiple projects. Depending on his or her experience, he or she consults on aspects of the system-level, architectural-level, or transistor-level design. Optionally works on the overall system and block specifications, and the test plan, with the team lead, the top-level designer, and system designer.

6.8 Program Manager

The program manager would coordinate with the team lead to manage the schedule and resources of the project. He or she would also manage the relationship with the “customer” as well has handle various financial responsibilities. Generally, the program manager manages multiple projects.

6.9 Behavioral Modeling

Responsibility for writing behavioral models varies depending on the complexity of the project and the type of the block. For simple blocks, the block designer usually writes the model. This is very practical because the model and its corresponding circuit share the same test bench. The block designer uses the model to efficiently construct and debug the test bench. Once the circuit is available, the block designer compares the behavior of the model against the block, and makes adjustments to improve its fidelity.

In this way, the designer uses his or her deep knowledge of the block and its performance to develop the model.

For more complex block like a phase-locked loop (PLL) or a clock-and-data recovery (CDR) circuits the behavioral models are often created by the system designer or by an assigned modeling engineer. Such models tend to call for sophisticated and rather specialized modeling skills and generally require substantial effort expended throughout the duration of the project.

The top-level designer would take responsibility for developing the behavioral models for overall system test bench, perhaps with the help of the system designer or modeling engineers.

7 Further Benefits of Top-Down Design

Besides the benefits described in the previous two sections, a rigorous top-down design methodology addresses all of the various needs and issues described in Sections 1-2.6, which includes the following.

7.1 Improves Communications Between Engineers

Communications between the designers is improved in two substantial ways. First, the use of a shared high-level model of the system that everyone verifies their designs in eliminates most of the miscommunication that occurs when following either bottom-up of primitive top-down design processes. In addition, the executable specifications and plans (models and scripts) are more specific and less ambiguous, and considerably reduce the time spent writing and reading formal specifications, providing a more efficient and effective replacement.

7.2 Improves Productivity

The improved productivity that results with a rigorous top-down design process is due mainly to the elimination of mistakes and respins. A more formal, less error-prone design process with better communication between engineers means that less time is spent making and recovering from mistakes, and more time is spent on productive design tasks.

7.3 Improves Ability to Handle Complex Designs

The ability of a design team following rigorous top-down design methodology to handle more complex designs follows from the better system exploration and from the increases understanding of the design that comes from it, and from the improved communications. In addition, the use of mixed-level simulation dramatically improves the teams ability to verify complex circuits.

7.4 Allows Parallel Execution of Design Tasks

Reducing time-to-market is an important way in which design teams can increase their chance of success and the returns of their products. Part of the reduction in time-to-mar-

ket is a result of the improved productivity and effectiveness of the design team, as described above. However, a rigorous top-down design methodology also has the benefit in that it allows more engineers to be effectively engaged in the development process at the same time, resulting in a further decrease in time-to-market.

As pointed out earlier, the existence of a shared executable high-level model of the system allows the test program development to be done in parallel with the block design and assembly, thereby eliminating a large contributor to the delay between when the design team and when manufacturing think the chip is ready to go. In addition, much of the final verification tasks that are needed in with a bottom-up design style are moved forward in the form of mixed-level simulation and performed in parallel by the block and top-level designers. The block developers can also get started developing models or evaluating IP while the system designer is finalizing the overall system architecture.

The improved and more formal communication that results in a rigorous top-down design methodology allows more engineers to be involved in the design process without overstressing the shared members of the team: the team lead and the top-level and system designers. There is also a natural support for hierarchy on large projects. Only two levels have been described in this paper, but a large chip can be partitioned into major sections (ex. RF, analog, digital, etc.), with overall leaders for the whole chip, as well as leaders for the individual sections.

7.5 Supports IP Reuse

Not only does the top-down design process described in this document improve the communication between the members of the design team, but when the design is being reused, it also improves communication between design teams. If the design is being sold, then the process also improved the communications between different companies: seller and buyer.

A rigorous top-down design process creates as a natural by-product a thoroughly validated high-level description of the design, which is a critical enabler of IP reuse. This description is used by potential customers when evaluating the IP and by customers when integrating the IP. To see the value of having the needed model fall out of the design process as a by-product, consider the case where it does not. Using a bottom-up design process requires that the model be developed after the design was completed. This creates several barriers to the success of the IP. First, with the model not being used as an integral part of the design process it does not get much in the way of incidental testing. Substantial extra effort is required to field a high quality model, resulting in extra cost and delay. Furthermore, it is unlikely that the same quality model would be developed with an adjunct process. Second, with the model not being leveraged during the design process, the total cost of developing the model offsets any revenue from the IP, requiring higher levels of market success to break even. Finally, the model development process delays the release of the IP. This is especially troublesome as the price of IP drops dramatically as it becomes a commodity. Time-to-market is especially critical in the IP market as the price can drop by a factor of ten within a year of its release. Delay of even a month dramatically affects the total revenue of a product.

8 Mixed-Signal Hardware Description Languages

Both Verilog-AMS and VHDL-AMS have been defined and simulators that support these languages are emerging. These languages are expected to have a big impact on the design of mixed-signal systems because they provide a single language and a single simulator that are shared between analog, digital, and eventually system designers. It will be much easier to provide a single design flow that naturally supports analog, digital and mixed-signal blocks, making it simpler for these designers to work together.

AMS languages make it substantially more straight-forward to write behavioral models for mixed-signal blocks, and they bring strong event-driven capabilities to analog simulation, allowing analog event-driven models to be written that perform with the speed and capacity inherited from the digital engines. This is very important, because most of the analog and mixed-signal models used in high-level simulations are naturally written using event-driven constructs. For example, blocks like ADCs, DACs, PLLs, ΣΔ converters, discrete-time filters (switched-capacitor), etc. are easily and very efficiently modeled using the analog event-driven features of the AMS languages.

Finally, 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 in the foreseeable future because the only synthesis that is available for analog circuits is very narrowly focused.

8.1 Verilog-AMS

Verilog-A is an analog hardware description language patterned after Verilog-HDL [16]. Verilog-AMS combines Verilog-HDL and Verilog-A into an MS-HDL that is a super-set of both seed languages [27]. 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 mixed-signal behavioral models. A unique feature of Verilog-AMS is that it provides 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 real-valued event-driven nets and for back-annotating interconnect parasitics.

AMS Designer, a commercial version of Verilog-AMS that also supports VHDL, is available from Cadence Design Systems [2].

8.2 VHDL-AMS

VHDL-AMS [8,18] adds continuous time modeling constructs to the VHDL event-driven modeling language [17]. Like Verilog-AMS, mixed-signal behavioral models can be directly written in VHDL-AMS. Unlike with Verilog, there is no analog-only subset.

VHDL-AMS inherits support for configurations and abstract data types from VHDL, which are very useful for top-down design. However, it also inherits the strongly typed nature of VHDL, which creates problems with mixed-signal designs. Within VHDL-AMS you are not allowed to directly interconnect digital and analog ports, and there is no support for automatic interface element insertion built-in to the language. In fact, you are not even allowed to directly connect ports from an abstract analog model (a signal flow port) to a port from a low-level analog model (a conservative port). This makes it

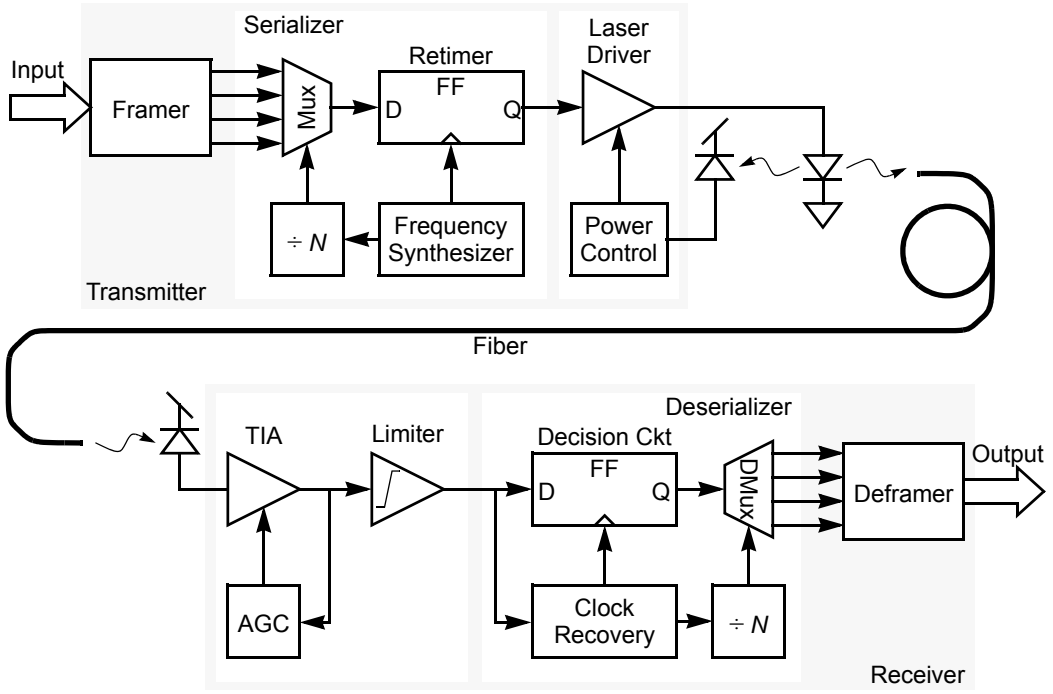
difficult to support mixed-level simulation. These deficiencies have to be overcome by a simulation environment, making VHDL-AMS much more dependent on its environment. This should slow deployment of effective VHDL-AMS-based flows.

ADvance-MS, a commercial version of VHDL-AMS that also supports Verilog, is available from Mentor Graphics [23].

9 Design Example

Recently a SONET OC-48 compliant serdes macro was designed by Cadence Design Systems. A serdes macro is an IP block intended to be instantiated on a large chip. Serdes is short for serializer-deserializer. It is illustrated in Figure 2. It is an essential part of a high-speed serial link and includes the whole link interface. The figure shows a unidirectional link with a serializer at one end and a deserializer at the other. In addition, there is always a reverse link (not shown). As such, each end of the link requires both a serializer and a deserializer (a serdes).

FIGURE 2 *An optical transceiver system.*



The deserializer is the more challenging of the two halves of a serdes, as it requires clock recovery. To get the highest data rate serial links use a non-return to zero (NRZ) coding in which the bit value is kept constant for the whole clock cycle. The NRZ coding produces a signal spectrum that contains little energy at the actual clock frequency, and so a clock recovery circuit is needed. It is typically a specialized type of phase-locked loop (PLL). When multiple serdes reside on the same chip, the voltage-controlled oscillators (VCO) found in most PLLs (see Figure 3) can inadvertently interact

with each other. Coupling that occurs between the oscillators can cause one to injection lock to another, which would result in a failure to recover the clock. Cadence avoided this problem by using a single reference oscillator, and by replacing the VCO in each PLL with a phase interpolator fed by the reference oscillator as shown in Figure 4.

FIGURE 3 *Conventional clock and data recovery.*

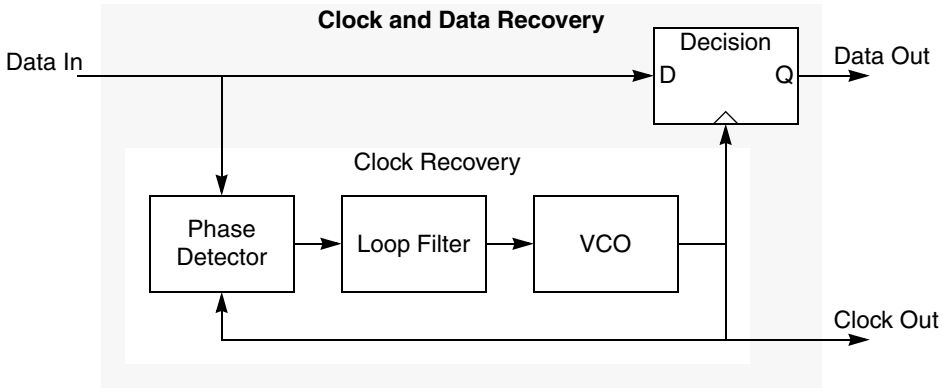
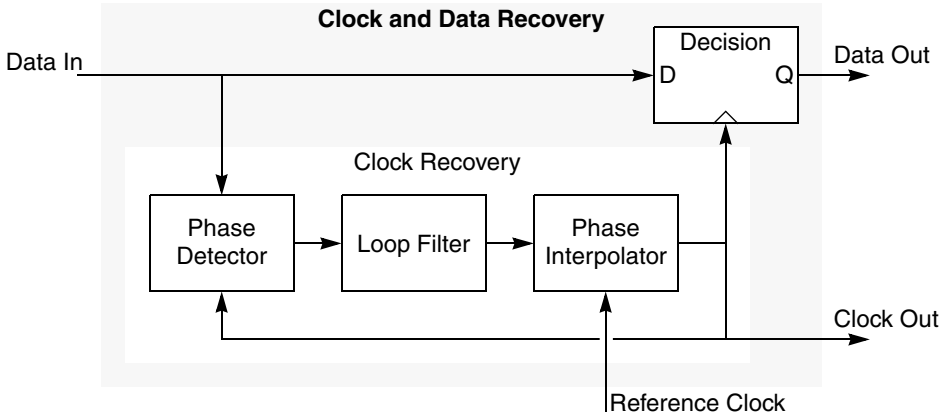


FIGURE 4 *Clock and data recovery using reference oscillator and phase interpolator.*



The design occurred before the design group adopted an AMS simulator, but given the largely digital nature of the circuit, they were able to use Verilog to support a mixed-signal top-down design process. After an initial algorithm exploration phase using models written in a general programming language, the architectural design phase commenced by writing Verilog models for every cell. The high-level architectural model was used to verify correct operation in each of the over 100 operating modes of the serdes. The main concern of the designers centered around the charge pump and phase interpolator, so a verification plan was developed that used mixed-level simulation to extensively verify the clock recovery circuit with these blocks at the transistor level. Designers were more comfortable with the purely digital components such as the phase detector and the divider, and so traditional cell characterization was used to develop bottom-up models that were included in most of the simulations. Mixed-level simulations were performed over a combination of nearly 100 corner cases in terms of process parameters, temperature, and supply voltage to assure reliable operation of the phase interpolator. Each sim-

ulation covered on the order of 2000 cycles of the high frequency clock. These mixed-level simulations were used to identify the worst corners, over which full transistor level simulations were the performed that included the entire clock recovery circuit.

The verification plan also specified a series of tests to confirm performance in demanding situations. For example, the reaction of the circuit was observed while being subjected to noise or jitter from various sources, such as at the input, through the bias and supply lines, and from the reference oscillator. In addition, it specified a series of challenging situations for which operation had to be verified, such as when the input contained long streams of zeros or ones that are allowed under SONET standards, which with a NRZ code results in long intervals during which no input transitions are received, making lock difficult to maintain. With the phase interpolator at the transistor level (about 1000 transistors) simulations took 3-6 hours, and with the entire clock recovery circuit at transistor level, the simulations required 3-4 days. Thus, a full transistor level simulation required the same compute resources of roughly 30 mixed-level simulations.

As a reusable block, an important part of the output of the design process is a model that can be used both to evaluate the block to determine its suitability for use in a larger system, and as part of the system integration and verification process. The Verilog model that was developed and refined during the block design phase is a natural choice, as it has been extensively tested and refined. However, as it is a Verilog model, it cannot model the degradation in the performance of the system as a result of the parasitics from the package, the board, or the connectors (the Verilog model allows timing-related impairments to be easily modeled, but not impairments due to effects such as dispersion, variations in the input signal amplitude, small couplings, etc.). For this reason, another model is provided that describes the front-end of the serdes (the input amplifier and equalizer) in a SPICE format. The system integrators are expected to use this model to assure that the input signal reaches the deserializer with enough fidelity (has a sufficiently open eye) to assure proper operation.

While in this case it was possible to follow a top-down design methodology using an older mixed-signal simulator that provides a relatively distant coupling of SPICE and Verilog, the design process could be substantially improved by using Verilog-AMS. Virtually all of the behavioral modeling for the cell was done using Verilog, which means that no analog effects could be included. This was problematic in several situations. The first of which is that two models needed to be developed and delivered to the customer. A single Verilog-AMS model would have been sufficient to include both the complex event-driven behavior of the deserializer, as well as the analog behavior at the input, including the degradation that occurs as a result of parasitics from the package, the board, and the connectors. In addition, both the equalization in the input amplifier and amplitude and pre-emphasis levels in the driver are programmable, a fact that had to be ignored when the model was delivered in two distinct non-interoperable pieces, but could have been easily included in a single Verilog-AMS model. Finally, use of Verilog-AMS would have allowed more analog effects to be modeled during the mixed-level simulation process. For example, in a multi-lane serdes macro there can be up to 200 bias current lines distributed to the many lanes. These bias lines were not included in the top-level model because of limitations in Verilog. They could have been included if Verilog-AMS were used. Mis-connection of any of the bias lines would have caused the associated lane to fail.

10 Conclusion

Many design groups currently claim to be following a top-down design process, yet experience most of the problems attributed to the use of a bottom-up design style. This is because they are basically employing a bottom-up style with a few relatively cosmetic changes that serve to give the appearance of top-down design. This paper lays out a series of principles that must be followed to realize all of the benefits associated with a rigorous top-down design methodology.

A rigorous top-down design methodology requires a significant investment in time and training and a serious commitment throughout the design process if it is to be successful. However, it is much easier the second time around and once mastered provides dramatic returns. Fewer design iterations and silicon respins are needed, which results in a shorter and more predictable design process. More optimal designs are produced that are better verified. It allows design teams to be larger and more dispersed, giving the option of trading a higher initial investment rate for a shorter time-to-market. And it is relatively tolerant of changes in the requirements that occur late in the design cycle.

Employing a rigorous top-down design methodology dramatically increases the effectiveness and productivity of a design team. If a design team fails to move to such a design style while its competitors do, it will become increasingly ineffective. It eventually will be unable to get products to market in a time of relevance and so will be forced out of the market.

Given the high pressure world that most designers live in, it is difficult for them to acquire the skills needed to be successful in a rigorous to-down design methodology. In addition, there is little training available from continuing education centers. This suggests that the transition to top-down design will be slow. The best hope for accelerating the move to top-down design is for universities to give designers the necessary background and training in the benefits and practice of rigorous top-down design. There are some signs that this is beginning [12], but it is not as aggressive or as widespread as it needs to be in order for there to be a smooth and timely transition.

10.1 If You Have Questions

If you have questions about what you have just read, feel free to post them on the *Forum* section of *The Designer's Guide Community* website. Do so by going to *www.designers-guide.org/Forum*.

Acknowledgments

Much of the material presented is based on discussions with Dan Jefferies, Henry Chang and Alessandro Piovaccari of Cadence Design Systems and Jim Holmes of Texas Instruments.

Bibliography

[1] B. G. Arsintescu, E. Charbon, E. Malavasi and W. Kao. AC constraint transformation for top-down analog design. *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems* (ISCAS '98), vol. 6, 1998, pp. 126-130.

[2] Cadence Design Systems. AMS Designer, *www.cadence.com*.

[3] Cadence Design Systems. Signal Processing Worksystem, *www.cadence.com*.

[4] N. Chandra, and G. W. Roberts. Top-down analog design methodology using Matlab and Simulink. *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems* (ISCAS '01), vol. 5, 2001, pp. 319-322.

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

[6] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, L. Todd, *Surviving the SoC Revolution*, Kluwer Academic Publishers, 1999.

[7] E. Chou and B. Sheu. Nanometer mixed-signal system-on-a-chip design. *IEEE Circuits and Devices Magazine*, vol. 18, no. 4, July 2002, pp. 7-17.

[8] E. Christen, K. Bakalar. VHDL-AMS — a hardware description language for analog and mixed-signal applications. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 10, Oct. 1999, pp. 1263-1272.

[9] Jeremy Donovan. The truth about 300 mm. *Electronic Engineering Times*, November 18, 2002.

[10] C. Force, T. Austin. Testing the design: the evolution of test simulation. *International Test Conference*, Washington 1998.

[11] C. Force. Integrating design and test using new tools and techniques. *Integrated System Design*, February 1999.

[12] J. E. Franca. Integrated circuit teaching through top-down design. *IEEE Transactions on Education*, vol. 37, no. 4, Nov. 1994, pp. 351-357.

[13] G. G. E. Gielen. Modeling and analysis techniques for system-level architectural design of telecom front-ends. *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 1, part 2, Jan. 2002, pp. 360-368.

[14] G. G. E. Gielen. System-level design tools for RF communication ICs. URSI International Symposium on Signals, Systems, and Electronics (ISSSE '98), 1998, pp. 422-426.

[15] J. Holmes, F. James, and I. Getreu. Mixed-Signal Modeling for ICs. Integrated System Design Magazine, June 1997.

[16] IEEE. *Standard Description Language Based on the Verilog™ Hardware Description Language,* IEEE Standard 1364-1995.

[17] IEEE. *VHDL Language Reference Manual,* IEEE Standard 1076-1993.

[18] IEEE. *Definitions of Analog and Mixed-Signal Extensions to IEEE Standard VHDL.* IEEE Standard 1076.1-1999.

[19] K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, and F. Sendig. Design of mixed-signal systems-on-a-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, Dec. 2000, pp. 1561-1571. Available from *www.designers-guide.org*.

[20] Kenneth S. Kundert. *The Designer's Guide to Verilog-AMS*. Kluwer Academic Publishers, 2004.

[21] E. Malavasi and E. Charbon. Constraint transformation for IC physical design. *IEEE Transactions on Semiconductor Manufacturing*, vol. 12, no. 4, Nov. 1999, pp. 386-395.

[22] Mathworks. Matlab and Simulink, *www.mathworks.com*.

[23] Mentor Graphics. ADVance MS simulator, *www.mentor.com*.

[24] T. Murayama, Y. Gendai. A top-down mixed-signal design methodology using a mixed-signal simulator and analog HDL. *Proceedings EURO-DAC '96, The European Design Automation Conference and Exhibition*, 1996, pp. 59-64.

[25] *Verilog-AMS Language Reference Manual: Analog & Mixed-Signal Extensions to Verilog HDL*, version 2.1. Accellera, January 20, 2003. Available from *www.accellera.com*. An abridged version is available from *www.verilog-ams.com* or *www.designers-guide.org/VerilogAMS*.

[26] H. Samueli. Broadband communication ICs: enabling high-bandwidth connectivity in the home and office. Keynote address in the *Slide Supplement to the Digest of Technical Papers for the 1999 International Solid State Circuits Conference*, pp. 29-35.

[27] Teradyne. SpectreVX and SaberVX virtual test environments, *www.teradyne.com*.