Top-Down Design of Mixed-Signal Circuits

Ken Kundert

Cadence Design Systems
San Jose, California, USA

Abstract

With mixed-signal designs becoming more complex and time-to-market windows shrinking, designers cannot hope to keep up unless they change the way they design. They must adopt a more formal process for design and verification: top-down design. It involves more than simply a cursory design of the circuit block diagram before designing the blocks. Rather, it also involves developing and following a formal verification plan and an incremental and methodical approach for transforming the design from a abstract block diagram to a detailed transistor-level implementation.

1. Introduction

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 time more effort per transistor than designing digital control logic, though this factor was normalized out of the 14× ratio.

In my experience, the primary culprits behind the poor productivity of those at the bottom of the scale are increasingly complex designs combined with a continued preference for bottom-up (i.e., transistor-level) design methodology and the occurrence of simulation late in the design cycle, which leads to errors and re-spins. There's a huge disparity in productivity between those mixed-signal designers who have transitioned to a "top-down" design methodology and use mixed-signal hardware description languages (MS-HDLs), and those who practice "bottom-up" design and rely solely on SPICE.

1.1. 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 more quickly 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. To understand this, consider three scenarios for developing a product, illustrated in Figure 1. First consider

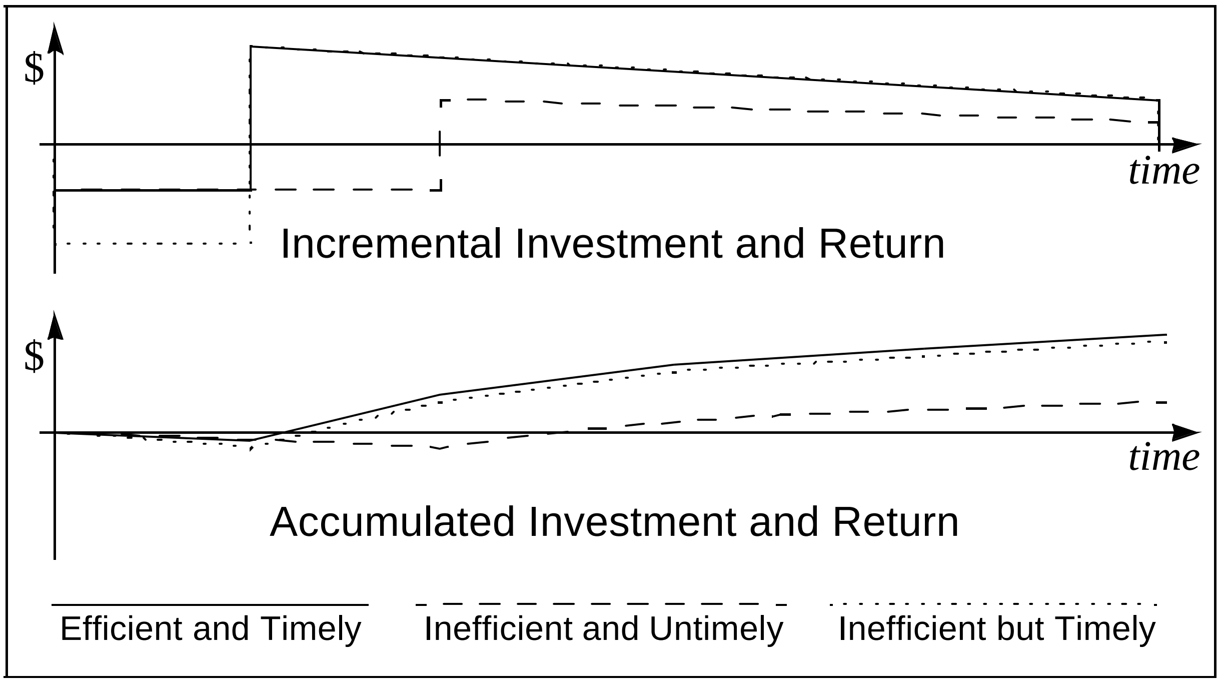

Figure 1: The expected investment and return for the same product developed using three different approaches.

employing an efficient product development process and being first to market. Then consider using the same number of developers with an inefficient development process. Furthermore, assume that this causes the product to be late to market, which 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 would be 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 shows that it is 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 very valuable capability. With follow on products, the situation is somewhat different. Here, 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.

1.2. Design Productivity Gap

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 either 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. 30 years ago a typical receiver contained between 5 and 10 transistors whereas it is common for a modern cell phone to contain 10M transistors. Second, the operation of the circuits are becoming more complex. 30 years ago integrated circuits generally consisted of simple functional blocks such as opamps 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 thousands of cycles. Examples include PLLs, ΣΔ converters, magnetic storage PRML channels, and CDMA transceivers. The result of these two effects together is that complexity is increasing at a blistering pace, and is outstripping the designers ability to keep up. For example, 30 years ago you could build a radio from a rock and a wire, whereas to build a modern radio requires more compute power than existed in the largest supercomputer available back then.

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. 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. If a design group fails to move to such a design style, 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.

2. 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. At this point, 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.

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

- 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.
- 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.
- Any errors or problems found when assembling the system are expensive to fix because they involve redesign of the blocks.
- 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.
- 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. 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. In a basic top-down approach, the architecture of the chip is defined as a block diagram and simulated and optimized using either a MS-HDL simulator or a system simulator. 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 in. And while this is a step towards top-down design, it only addresses one of the issues with bottom-up design and there is much more that can be done. To overcome the remaining issues, one must go further. Also needed is a formal verification planning procedure and a mixed-level verification strategy. Both act to reduce risk and are added with the understanding that with complex mixed-signal circuits, complete final transistor-level verification is at a minimum very expensive and is often impractical. With careful planning one can use mixed-level simulation to move the verification up in the design process where it is less expensive and so can be more comprehensive. It also tends to find errors earlier in the design process when recovery is easier and less expensive.

A well designed top-down design process methodically proceeds from architecture- to transistor-level design. Each level is fully designed before proceeding to next and each level is fully leveraged in design of next. It 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 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 would already be available and can be quickly applied to verify any changes.

3.1. System Architect

The system architect is a new member of the design team. He or she is the leader of the top-down design process and is expected to develop the simulation and modeling plans and to coordinate with the other designers to assure that the plans are followed. The primary responsibility of the system architect is to assure that the system operates as expected when finally implemented. This must be a designer that has experience in the type of system being designed so that he or she can anticipate and plan for issues that are likely to occur. Preferably, the experience covers aspects of both system and block design. The system architect may be the system engineer, the one that actually designs the block diagram, but it need not be. However, the system architect should not also have block design responsibilities. Block design has a tendency of consuming an engineer.

The system architect should be comfortable with modeling and MS-HDLs to the point where he or she can write the descriptions of both the system and the individual blocks. Since most designers are not skilled at modeling and not familiar with MS-HDLs, the system architect should also be able to train the other engineers on the project on the use of MS-HDLs. However, it is important to recognize that modeling is a skill that is distinct from design. Designer often have neither the skill nor the inclination to write sophisticated models. If sophisticated models are required, they generally must be developed by the system architect.

The system architect must have a good understanding of simulation. In particular, where and how simulation can be used to verify areas of concern in the design, and where it cannot. This knowledge is leveraged heavily during the development of the simulation plan.

The system architect 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.

Once the top-level schematic is captured, the top-level models are written, usually by the system architect, and the system completely verified according the simulation plan. The top-level schematic and models are then distributed to everyone on the design team. As the design progresses, the system architect would approve and coordinate any changes to the block interfaces, and then distribute updated models of the system or the blocks to the team. As the block designers work, they would provide transistor-level schematics (pre- and post-layout) to the system architect, who would then verify them with mixed-level simulation, again according to the simulation plan, before accepting them.

During the design process, the system architect would work with the test engineers 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 [1,2,3,6].

3.2. Simulation and Modeling Plans

An important focus in a good top-down design methodology is the development of a comprehensive simulation plan, which in turn leads to a modeling plan. This is done by the system architect with input from the whole design team. 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 would specify how the test would be preformed, and which blocks would be 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. These models may be written either by the system architect or the block designer.

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. Following these general rules will result in faster simulations and less time spent writing models.

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. The simulation and test plans would initially be applied to the high-level description of the system, where they can be quickly debugged. Once available, they can be applied during the mixed-level simulations of the blocks, reducing the chance that errors will be found late in the design cycle.

3.3. System-Level Verification

System-level design is generally performed by system engineers. Their goal is to find an algorithm and architecture that implement the required functionality while providing adequate performance at minimum cost. They typically use system-level simulators, such as Matlab, Simulink or SPW [4,5], 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, they support the abstract data types used at the system level, and 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. 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 [6] or VHDL-AMS [8] 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 just now becoming available. It will probably take a while for this approach to become established because of the absence of application specific libraries.

3.4. Mixed-Level Verification

Digital synthesis maps digital behavior onto digital gates that are arranged in a rather constrained topology. The simple nature of gates combined with the constrained topology makes synthesis feasible. With analog circuitry, the fundamental building blocks are much more complex and varied and the topology is completely unconstrained. These two factors make analog synthesis a fundamentally much more difficult problem than digital synthesis. Analog synthesis so far has resisted all attempts at automation except in limited cases, such as analog filters. Work continues, but we are still far from having universal analog synthesis.

Without analog synthesis, analog design is done the old fashioned way, with designers manually converting specifications to circuits. While this allows for more creativity, 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 will 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. In mixed-level simulation, the system, described at a high level, acts as a test-bench for the block, which is described at the transistor level. Thus, the block is verified in the context of the system, and it is easy to see the effect of imperfections in the block on the performance of the system. Mixed-level simulation 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 by the system architect.

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 verify most mixed-signal systems.

3.5. Bottom-Up Verification

Once a block is implemented, one could update the models that represent it to more closely mimic its actual behavior. This would improve the effectiveness of mixed-level and system-level simulation. This process 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.

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.

Though bottom-up verification is helpful when verifying the performance of large systems, it is rarely done today. Generating behavioral models that include the detailed behavior of even simple blocks is quite difficult and requires a specialized skill not commonly found in the design team. This situation is not expected to change until automated tools and methodologies develop to generate detailed behavioral models.

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

3.6. Final Verification

In a top-down design process, SPICE-level simulation is used judiciously in order to get the benefits without incurring the 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 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.

4. MS-HDLs

Both Verilog-AMS and VHDL-AMS have been defined and commercial implementations 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 and digital 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. It also becomes substantially more straight-forward to write behavioral models for mixed-signal blocks. Finally, the AMS languages 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.

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

4.1. Verilog-AMS

Verilog-A is an analog hardware description language patterned after Verilog-HDL. Verilog-AMS combines Verilog-HDL and Verilog-A into a MS-HDL that is a super-set of both seed languages [7]. 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. Verilog-AMS also 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 back annotating interconnect parasitics.

A commercial version of Verilog-AMS that also supports VHDL is expected soon from Cadence Design Systems.

4.2. VHDL-AMS

VHDL-AMS adds continuous time modeling constructs to the VHDL event-driven modeling language [8]. 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 both the good and the bad aspects of VHDL. For example, VHDL-AMS inherently supports configurations and abstract data types. However, VHDL is also strongly typed, which is a serious problem for mixed-signal designs. You are not allowed to interconnect digital and analog ports, and there is no support for automatic interface element insertion. In fact, you are not even allowed to connect ports from an abstract analog model (a signal flow port) to a port from a low-level analog model (a conservative port). This makes it difficult to support mixed-level simulation. VHDL-AMS also does not provide support for back-annotation of RC interconnect. These represent fundamental flaws that will have to be overcome by a simulation environment, making VHDL-AMS much more dependent on its environment. This should slow deployment of effective VHDL-AMS-based flows.

A commercial version of VHDL-AMS that also supports Verilog is available from Mentor Graphics [9]. A VHDL-AMS simulator is also expected soon from Analogy [9].

5. Example

Though this example is several years old, it is representative of the type of circuit complexity that is becoming mainstream today. It is a PRML channel chip that it 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 would be 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.

6. Development of System Architects

The primary barrier to widespread adoption of a top-down design style for complex mixed-signal circuits is a lack of engineers with the skills and training to be system architects. A system architect must

- Be fluent in an AMS language and skilled in the art of modeling
- Be an experienced designer
- Have a good understanding of the top-down design process
- Be proficient in the use of circuit and AMS simulation
- Have the ability to lead and manage complex projects

Given the high pressure world that most designers live in, it is unlikely that they will be able to acquire such a broad and deep set of skills while on the job, even if they are motivated to do so. Rather, it is important for their employers to look for engineers that have the interest and the relevant background and invest the time and training to develop them into system architects. In addition, it is essential that appropriate training becomes available from universities and continuing education centers.

7. Conclusion

Top-down design is a formal design process that requires a serious commitment throughout the entire design process. It is not a piece of software or something you do in your spare time. It is not a way to reduce headcount or something you can try after the design is complete. It is considerably more than simply doing the initial design of the block diagram with Simulink and it is not something you can be successful at without a significant investment in time and training. However, it is much easier the second time around and once mastered provides a dramatic return. Using top-down design usually results in needing fewer design iterations, which provides a more predictable design process. It also results in more optimal designs in a shorter time. Finally, it allows design teams to be larger and more dispersed, giving the option of trading a higher initial investment for a shorter time-to-production.

Acknowledgements

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

Bibliography

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

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

[3] Dantes virtual test environment, *www.virtualtest.com*.

[4] Matlab and Simulink, *www.mathworks.com*.

[5] *Signal-Processing Worksystem User’s Guide.* Cadence Design Systems, San Jose, CA.

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

[7] *Verilog-AMS Language Reference Manual: Analog & Mixed-Signal Extensions to Verilog HDL*, version 2.0. Open Verilog International, 2000. Available from *www.ovi.org*.

[8] VHDL-AMS, *www.vhdl.org/analog*.

[9] VHDL-AMS simulators, *www.vhdl-ams.com.*