Specification Driven Analog and Mixed-Signal Verification

Henry Chang, Ken Kundert
Designer’s Guide Consulting, Inc
101 First Street, #150
Los Altos, CA 94022

***Abstract*- We describe an analog/mixed-signal verification methodology that is practical, efficient, systematic, scalable, and repeatable. The methodology begins with analog block-level specifications and ends with analog block schematics that have been verified in a self checking manner against the specifications using transistor level simulation, validated models of the analog that will be used as part of the chip-level verification process, and self-checking regression tests that validate that the models match the schematic. Combined with digital verification, we will explain how the entire chip including the analog schematics is functionally covered. This approach find bugs in the analog schematics, the analog/digital interface, and the digital/embedded software that talks with the analog. We will also touch upon when digital verification techniques apply and when they do not.**

I. Introduction- The Analog/Mixed-Signal Verification Challenge

A. *The Market Challenge*

In the world of consumer systems, in order to compete and to remain profitable, system companies require getting innovative new products out-to-market on a regular basis. As a result, system companies are driving the semiconductor and fabless semiconductors companies into putting more functionality on integrated circuits (IC) and to do so in a shorter amount of time. For system companies to succeed, they require that semiconductor companies deliver parts on schedule to the specifications promised. Especially critical to system companies is that they receive first functional silicon for every integrated circuit “socket” in their system, so that they can test and work on the software that runs on their systems. Any one failure in an initial version of a chip could delay the system company from getting their product to market.

Almost all of these systems today have a significant amount of analog/mixed-signal content—most notable are the audio, touch screen, radio frequency (RF) transceivers, screen drivers, and motion detection components, but behind the scenes power management is playing a larger and larger role. Traditionally, the focus of analog/mixed-signal design is on performance. Analog designers focus on designing blocks to meet performance specifications such as signal-to-noise ratio, linearity, total harmonic distortion, bit error rate, and power supply rejection ratio. Designers do so through creative and innovative transistor level topologies, thorough system analysis, and a tremendous effort in simulating their designs across process, temperature, and voltage variations. Their typical tool set includes their schematic capture tool, their circuit simulator, and their simulation environment.

Today, analog/mixed-signal designs in these consumer integrated circuits are found in the context of medium to large systems characterized by a large amount of digital control including custom RTL and microprocessors/microcontrollers with embedded software/firmware. These ICs are used in flexible ways, and the system companies want to be able to program how they are used. Plus, in order to achieve higher levels of analog performance, it has been discovered that clever combinations of digital logic and analog circuitry can yield better performing analog designs.

In the world of consumer electronics today, if one were to ask a system company, what is more important in a first piece of silicon—function or performance, they would likely say function. For example, is it more important that the system software be able to control the audio output from the device, or is it more important that the audio output be CD quality? The system company would probably say the former. After all, if the system can't control the audio output, what good is the performance on the output?

Why are chips failing to achieve first functional silicon? There is an increasing dependency between analog and digital, and the co-verification of the interfaces is half-hearted. Bugs are found on both sides of interfaces—incompatibility between analog and RTL, incompatibility between analog and software, and incompatibility between analog blocks.

B. *What is Analog/Mixed-Signal Verification?*

First and foremost, the focus of analog/mixed-signal verification is functional verification. This is distinct from analog performance analysis which is what analog designers traditionally spend most of their time doing. Since achieving ever increasing performance targets is still necessary, analog/mixed-signal verification is an *additional* step needed on top of what is done by traditional analog designers.

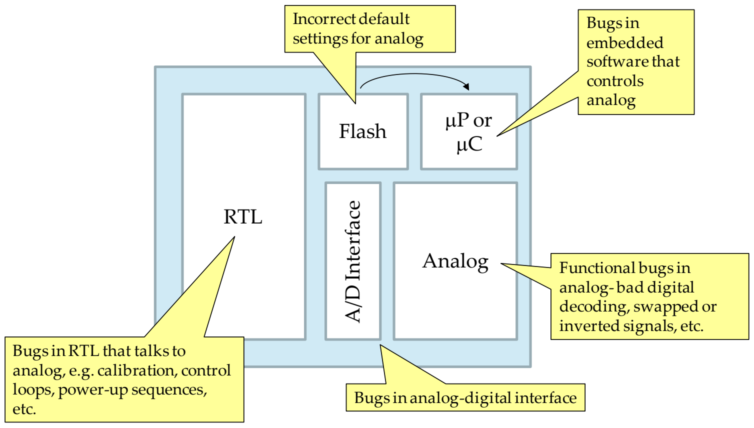

Incorrect default settings for analog
Bugs in embedded software that controls analog
Flash
μP or μC
RTL
A/D Interface
Analog
Functional bugs in analog- bad digital decoding, swapped or inverted signals, etc.
Bugs in RTL that talks to analog, e.g. calibration, control loops, power-up sequences, etc.
Bugs in analog-digital interface

Figure 1. Analog/Mixed-Signal Verification

Figure 1 illustrates what is included in analog/mixed-signal verification. Analog/mixed-signal verification looks for all bugs in the analog schematics, e.g. bad digital decoding, swapped or inverted signals. It looks for bugs in the analog-digital interface, such as missing level shifters, or unconnected signals. It looks for bugs in the RTL that talks to the analog, e.g. calibration, control loops, power-up sequences, and it looks for incorrect settings in the flash that runs on embedded microprocessors or microcontrollers.

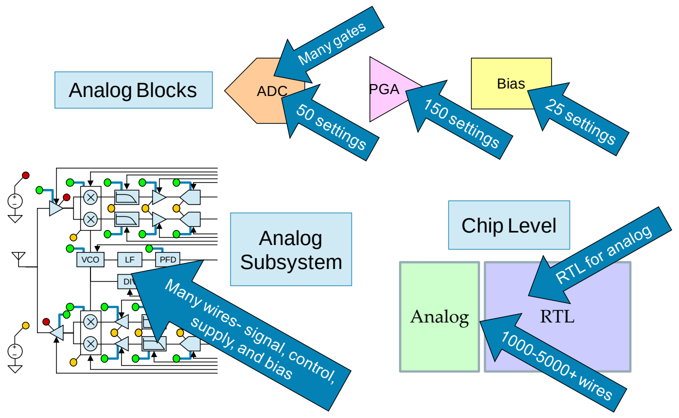


Figure 2. Complexities in Analog/Mixed-Signal Verification

The complexity in analog/mixed-signal verification lies at many levels, as shown in Figure 2. At the analog block-level, blocks have many modes of operation such as different power modes, and many degrees of tuning or trimming for performance, e.g. resistors, capacitors, bias sources/sinks, common mode voltages can often be trimmed. At the analog subsystem level where the analog blocks are connected, there are many wires, including all of the supply lines, current bias lines, voltage references, and digital control lines. All wires need to be connected correctly for the IC to be fully functional. Common errors include wiring blocks to an incorrect supply domain, incorrectly connecting bias lines so that one bias output goes to more than one bias input, and incorrectly connecting a bias input to another bias input. At the chip-level, there are thousands of wires that go from the digital domain to the analog. All of these need to be checked to be certain that they are connected correctly.

C. Why are These Bugs Present and Why do They Go Undetected?

Unfortunately, the focus, skills, and tools used to validate performance in analog/mixed-signal design are not the same as the ones needed for analog/mixed-signal verification. The focus on validating performance is to analyze signals going through a main set of transistors often referred to as the signal path. For this, the traditional schematic testbench is appropriate and highly effective as demonstrated by ICs meeting higher and higher performance specifications, e.g. higher data rates in RF, better touch screen performance, and better power control. The focus of verifying function is verifying what surrounds this main set of transistors, i.e. all of the scaffolding circuitry with all of the digital switches that control, provide supplies, and biases that enables the main signal path. It is this focus where a schematic based testbench struggles. For example, it is cumbersome to use drawings to specify sequences of digital signals or to discuss coverage.

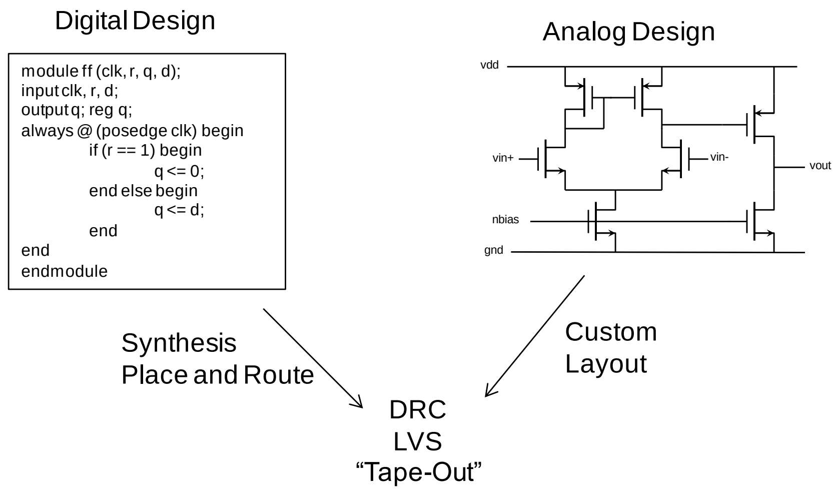
Further, schematic simulations are slow compared to simulating digital RTL and synthesized gate level logic. For chip-level simulation, behavioral models for the analog blocks are often needed to speed up simulations. Writing models is a time consuming, skill intensive, and error prone process that many analog designers are unwilling to do, or cannot do due to time constraints. As a result, at the chip-level, either only minimal simulation is done with the analog schematics, or simulations are done at the chip-level with low quality models. If models are not written, even if schematics could simulate faster, schematics are typically not finished until the end of the design process, so are not available in time for digital design and chip-level verification.

D. What It Takes to Achieve First Functional Silicon?

It takes a team. Design times of four months are not atypical. Tasks must be done in parallel. One cannot wait for analog schematics to start verification. Digitally corrected analog requires verifying the analog and digital together. Digital designers need functional models of the analog so that they can begin writing their RTL. Everyone benefits from the early availability of models—system designers, product engineers, test engineers, digital designers, digital verification engineers, and analog verification engineers. Analog schematics are too late and too slow for this team effort. Models are required.

E. How Does the Analog/Mixed-Signal Verification Challenge Differ from Digital?

Analog verification differs from digital verification in two key aspects. The first is that unlike digital design, analog design occurs at the *transistor level* in the form of schematics, not at the model level as shown in Figure 3.



```
module ff (clk, r, q, d);
input clk, r, d;
output q; reg q;
always @ (posedge clk) begin
        if (r == 1) begin
                q <= 0;
        end else begin
                q <= d;
        end
end
endmodule
```

Figure 3. Chip Design Flow

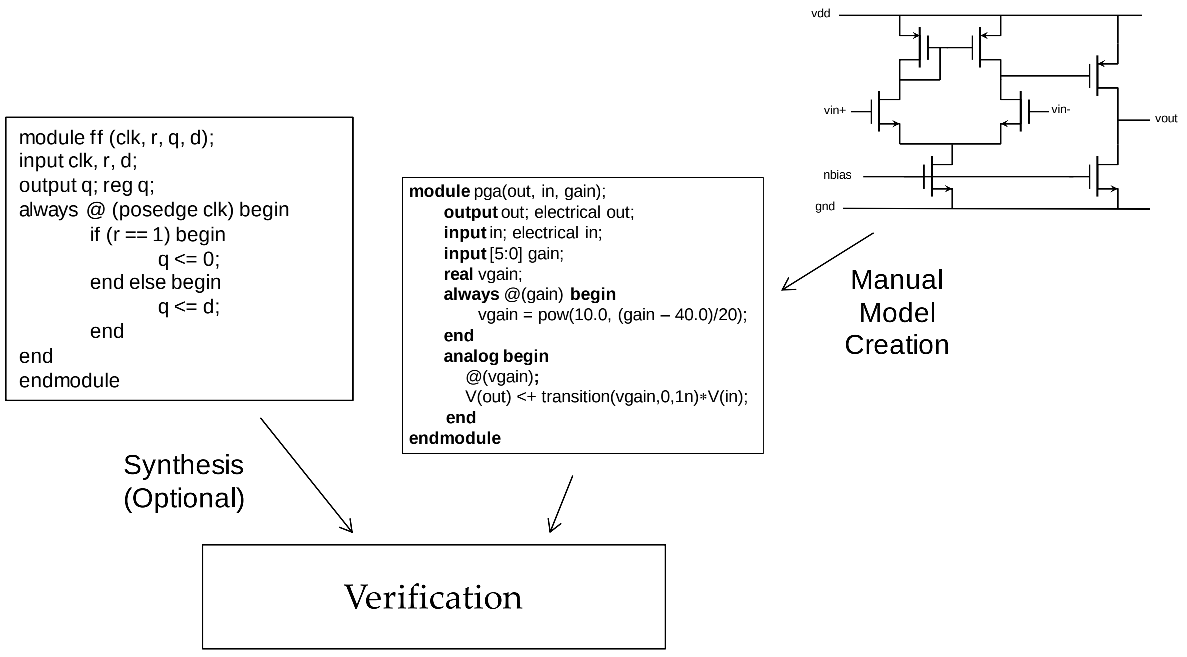
About a decade ago, there were two styles of design when connecting digital control to analog. One style was to have a narrow interface between digital and analog. This would minimize routing at the chip-level, but the burden was left in the analog subsystem to have digital decoding logic and to maintain state. A competing style was to have a wide interface between digital and analog, where the digital would take care of all things digital, i.e. the analog would contain little or no decoding logic and no state. This latter style has prevailed, largely because analog designers without easy access to digital tools such as static timing analysis and synthesis in their schematic based design environment, tended to make logic and timing errors. Having digital in the analog block also made determining fault coverage difficult. This leads to the second difference, being that the analog tends to have little in the way of digital state.

The first difference, analog needing to be designed at the transistor level, presents two key challenges for mixed-signal verification, while the second, the lack of state in the analog, helps make analog/mixed-signal verification feasible. The only known way to overcome the transistor level slowness and timeliness issues is to write behavior models for the analog blocks. Because writing models is not part of the current typical analog design process, additional resources and skills are needed. Models that are efficient, and models that tie into the digital flow are required. Writing models is the first challenge. Even when this is overcome, however, there is a more daunting challenge. When one writes a model, how does one assure that that model written matches the schematics? Unlike in digital, there is no equivalence checking to verify that the model written matches the schematic that is being fabricated. Chip-level verification assumes validated models. Without validated models, the chip-level verification becomes suspect.

With respect to writing analog block models and verifying that these models are equivalent to schematics, the fact that digital synthesis exists and that digital contains complex state leads to different verification approaches for digital compared to analog. Having complex state makes specifying digital designs much more difficult. With synthesis, implementation is straightforward. Because the specification is complex, and may be challenging to interpret, digital design verification requires two pairs of eyes to make certain that the translation from specification to RTL is correct. This leads to the strict separation between design and verification. Verification focuses on handling complexity and state. It is difficult to figure out how to get to specific states. This leads to concepts such as directed random, and much analysis is done to see that all states are visited (coverage). This leads to the assertions in testbench (monitor) approach. Since at any time a new set of vectors may be applied to a block due to directed random stimulus, one should take advantage of this by always checking that the operation of the block is correct. These assertions serve as the "second pair of eyes." Digital does not have an implementation problem (synthesis takes care of this). Digital has a specification interpretation and verification problem.

In analog, the issue is *not* state. Generally analog blocks have little state (if any) and the amount of state is decreasing over time as a result of applying the design style where all digital stays with the digital designers. Typically, for functionality at the analog block level, all states can be simulated ***exhaustively*** even at the transistor level, giving 100% coverage for the digital inputs. To a large extent, this eliminates the need for digital directed random, digital coverage analysis, and the need to place assertions (monitors) in the testbench for analog blocks. In fact, the job of placing the assertions in the testbench for analog blocks does not solve the most pressing problem, because the assertions placed in the testbench assumes a specification interpretation problem. Analog specifications are usually interpreted correctly. The focus for analog verification is to validate that the schematic is correct, i.e. that the specification was translated correctly to a schematic, because analog synthesis does not exist. Thus, the proper place to put the tests is in the testbench that simulates the schematic and in the testbench that validates that the model and schematic are equivalent. Once the tests are run, because the tests are exhaustive, we can now trust that the model has been verified to functionally match the schematics. If one were to place the assertions in the testbench surrounding the analog block model, that could act as a *third* check on the functionality, but unless the schematics are simulated at the chip-level, which they generally are not, this approach of putting the assertions in the testbench misses verifying the analog schematics. The "second pair of eyes" for the analog is verifying that the two independently created representations of the design, the schematic and the model, are equivalent. This will address bugs due to the analog specifications not being interpreted correctly.

At the analog block level, people do write specifications, but in most cases, the schematic is considered the *golden*. The creation of the schematic is a creative process where no correct-by-construction tools, such as synthesis, exists. Because schematics take most of the design time available and because schematics simulate slowly relative to digital RTL, the schematics are too slow to simulate at the chip level and are available too late in the design cycle to be used effectively as part of verification. Unlike digital, analog does not have a specification interpretation problem. The *ad hoc* analog specifications that exist are quite simple. Analog has a design creation (implementation) and verification problem. Further, it has a model creation problem. This is shown in Figure 4. Unlike digital where no additional effort is needed to simulate the design at the chip-level as far as the block level models go, the *majority* of the effort (as high as 80% [1]), is the creation of the verified analog functional models. Prior to any co-verification of analog and digital, the model creation effort must be invested before any verification can occur. For this majority of effort, digital techniques tend not to apply because analog is stateless. Also, the techniques that have made digital successful do not apply in solving the biggest problem in analog verification, the creation of these models, since in digital verification, the models exist.



```
module ff (clk, r, q, d);
input clk, r, d;
output q; reg q;
always @ (posedge clk) begin
        if (r == 1) begin
                q <= 0;
        end else begin
                q <= d;
        end
end
endmodule
```

```
module pga(out, in, gain);
    output out; electrical out;
    input in; electrical in;
    input [5:0] gain;
    real vgain;
    always @(gain) begin
        vgain = pow(10.0, (gain – 40.0)/20);
    end
    analog begin
        @(vgain);
        V(out) <+ transition(vgain,0,1n)∗V(in);
    end
endmodule
```

Figure 4. Chip Verification Flow

The closest analogy for the need to create new models is perhaps in the situation of simulating a microprocessor where simulating RTL is too slow, and a cycle approximate model is required. In the creation of that cycle approximate model, one must ask two questions. First, what information should be thrown away to make the simulation run faster, and second, how do we know that the cycle approximate model and the RTL are equivalent? The same two questions apply when writing an analog functional model.

F. The Analog Verification Blind Spots

Analog verification blind spots include analog designers often not understanding the need for models or that these models need to be fully verified. The need to do chip-level verification is not typically an analog designer's responsibility, as they can "tape-out" a chip just fine with only the schematics and no models. Analog designers also do not often understand that visual inspection is not verification. For digital verification engineers, they often do not understand that state is not the issue, and do not understand that the analog models were created by hand and are not the golden, i.e. the models are not what is being fabricated. The focus of analog verification needs to be on creating the models and validating that they are correct. An analog verification team should only be focused on directed random, looking at approaches to writing assertions in testbenches, and at higher level digital verification methodologies like UVM, only after the team has a solid analog model creation and model validation process, because without verified models, even the simplest of chip-level verification cannot be done.

II. The Analog/Mixed-Signal Verification Flow

A. The Overall Chip-Level Verification Flow

Chip-level verification is enabled by having verified analog models. Without analog verification to obtain those analog models, true chip-level verification simply cannot be done.

Once the challenges of creating analog block models are overcome, the analog block models can be plugged into a typical digital chip-level verification methodology in a relatively straightforward manner. Almost any digital chip-level verification flow can be used. The analog behavioral models can be in a language where only an event-driven simulator is required such as Verilog, VHDL, or SystemVerilog, or the models can be in a language where both an event-driven simulator and circuit simulator is required, such as Verilog-A, Verilog-AMS, or VHDL-AMS. Several simulator vendors have flows where the inclusion of the circuit simulator is seamless from the point-of-view of the digital event-driven simulator and the digital verification flow. Thus, all of the standard verification techniques (debugging, profiling, coverage analysis) can be applied to all of the digital and all of the digital in the analog models even when using a language such as Verilog-AMS.

B. Analog Block Level Verification Flow

Thus, verification with the analog hinges on efficient *validated* models of the analog blocks. The creation of these models should begin when the design begins. Models allow the co-design of digital control with analog. Self checking exhaustive testbenches are required to check that the model and the analog schematics are equivalent. Unvalidated models include models that have been written by the digital verification engineer, where no comparison is made with simulating analog schematics. These models were written based on the assumption that the paper specification is correct, but as mentioned, the schematic is the golden. There is no assurance that the models written match the schematics.

Another unvalidated approach is where the models are created by the analog designer or model engineer, where visual inspection of side-by-side simulation results are used to validate the models. This approach breaks down for two reasons. First, using the schematic based environment, it is non-trivial to simulate a block through all its modes and settings to exhaustively test both the model and schematic. And even if this could be done, in the typical simulation environment, it is non trivial to manually go through all of the simulation results to check that model and schematic simulation results are equivalent. Usually, this involves staring at waveforms. Second, since all of the above is mostly a manual effort, it is unlikely that an analog designer given the time pressures of finishing the design would be willing to repeat this process every time the schematic changes.

The validated model approach is shown in Figure 5 and is described in more detail in [2][3]. This flow is largely driven by an analog verification engineer. The skills required for this flow are different than the skills required for analog design or digital verification.

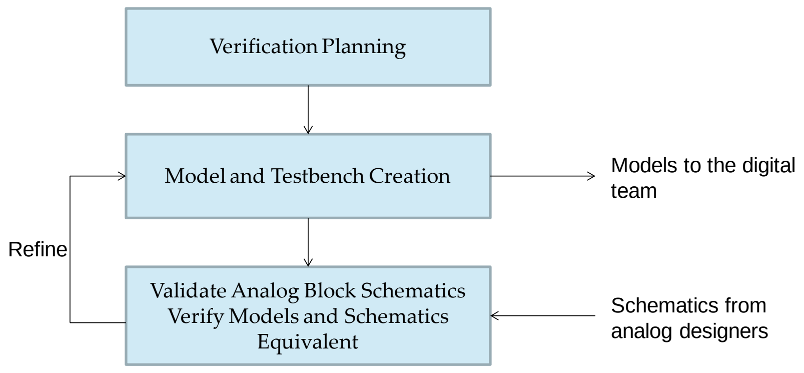
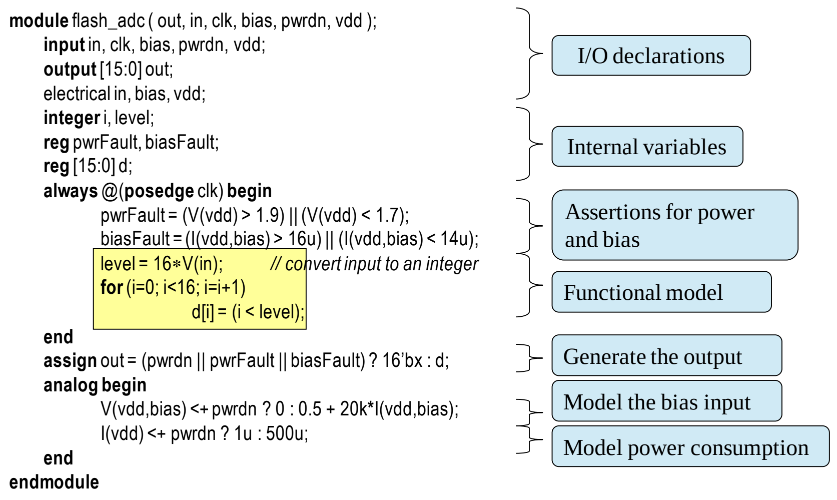

Verification Planning
Model and Testbench Creation
Models to the digital team
Refine
Validate Analog Block Schematics
Verify Models and Schematics Equivalent
Schematics from analog designers

Figure 5. Validated Model Development Flow

During the verification planning phase, with respect to the analog, the verification engineers determine what needs to be verified, what models need to be written, who will be using those models, how those models will be written in terms of language and at what level of abstraction. An overall estimate of the simulation resources required will also be determined. During the model and testbench creation phase, models are created, and self checking analog block level regression tests will be written based on what specifications exist. The models are shipped to the digital and chip-level team. Once the schematics are ready, the testbench is then applied to the schematics to validate that the schematics are correct and to validate that the model and testbench are functionally equivalent. Since, there’s little to no state in analog. Exhaustive functional testing can be done to assure full coverage of the schematic and to assure that the model and testbench are equivalent. This process is repeated as the schematics and/or specifications are updated.

C. Examples: Model and Testbench

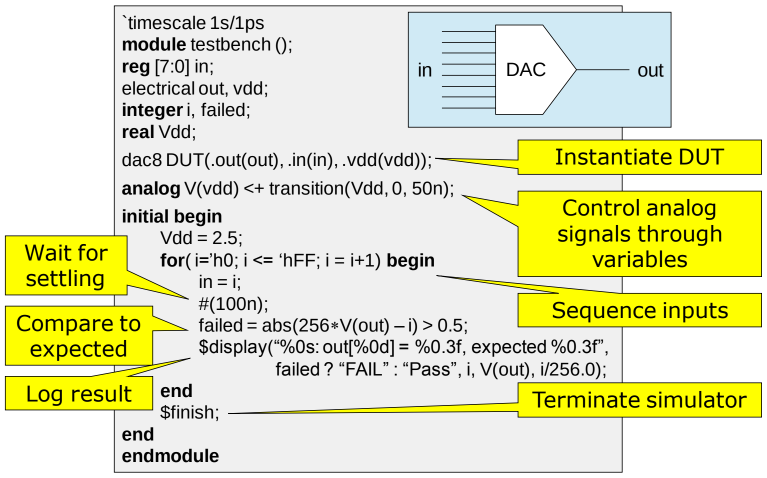
An example of a simple Verlog-AMS model of a flash analog-to-digital converter is shown in Figure 6. It illustrates the key features of the model. Ports that contain digital information are declared as Verilog wires or registers. In this example, these are “clk”, “pwrdn”, and “out”. Ports that contain analog information are declared as electrical signals. In this example, these are “in”, “bias”, and “vdd”. Wreal could also have been used to represent the analog. The use of wreal eliminates the need for the circuit simulator, but current/voltage behavior is lost which may be important on “vdd” and “bias” in this example. Depending on context, electrical models can be as fast as wreal models. Rather than model the output as a function of all of the inputs, we write assertions to check whether or not some of the inputs, such as “vdd” and “bias” are within the specified bounds. The behavior is written in an implementation neutral fashion, i.e. we model the behavior of the function of the block, not its architecture. Finally, the bias input load and approximate current consumption are modeled on the “bias” and “vdd” pins.



```
module flash_adc ( out, in, clk, bias, pwrdn, vdd );
    input in, clk, bias, pwrdn, vdd;
    output [15:0] out;
    electrical in, bias, vdd;
    integer i, level;
    reg pwrFault, biasFault;
    reg [15:0] d;
    always @(posedge clk) begin
        pwrFault = (V(vdd) > 1.9) || (V(vdd) < 1.7);
        biasFault = (I(vdd,bias) > 16u) || (I(vdd,bias) < 14u);
        level = 16∗V(in);        // convert input to an integer
        for (i=0; i<16; i=i+1)
                d[i] = (i < level);
    end
    assign out = (pwrdn || pwrFault || biasFault) ? 16'bx : d;
    analog begin
        V(vdd,bias) <+ pwrdn ? 0 : 0.5 + 20k*I(vdd,bias);
        I(vdd) <+ pwrdn ? 1u : 500u;
    end
endmodule
```

Figure 6. Key Features in a Verilog-AMS Model

An example of a simple self checking testbench of a digital to analog converter is shown in Figure 7. It illustrates the key features of the testbench. The testbench instantiates the device under test which could either be a model or the schematic. We run the testbench on the model, then on the schematic. If the testbench indicates that they both pass, then we conclude that the model and testbench are functionally equivalent. An alternative approach is to have a top-level netlist or schematic instantiate the testbench and the model, drive both with the same signals, and compare their outputs to see if they are within some tolerance. We should emphasize that whether one drives a block with a language based testbench or an equivalent schematic testbench (if it is possible), the simulation results will be the same. In the example testbench, shown in Figure 7, the electrical pins are driven in the Verilog-AMS analog block. Analog values are set in the “main” Verilog-AMS initial block. The sequences are applied to the device under test. For each input value in the sequence, the input is set. The testbench waits for a certain amount of time or for a condition to occur to allow the circuit to settle. It then compares the output to an expected value. It then writes the result including whether or not the test passed or failed to a log file. When the test sequence is complete, the simulation is terminated.



```
`timescale 1s/1ps
module testbench ();
reg [7:0] in;
electrical out, vdd;
integer i, failed;
real Vdd;
dac8 DUT(.out(out), .in(in), .vdd(vdd));
analog V(vdd) <+ transition(Vdd, 0, 50n);
initial begin
    Vdd = 2.5;
    for( i='h0; i <= 'hFF; i = i+1) begin
        in = i;
        #(100n);
        failed = abs(256*V(out) – i) > 0.5;
        $display("%0s: out[%0d] = %0.3f, expected %0.3f",
                    failed ? "FAIL" : "Pass", i, V(out), i/256.0);
    end
    $finish;
end
endmodule
```

Figure 7. Key Features in a Verilog-AMS Self Checking Testbench

A different example can be found at [4]. Please note that the example model and testbench in this paper are for illustrative purposes only. Though they will run, the key aspects shown require more code to work robustly in a production setting. A production model of this same example would be approximately 200 to 300 lines of code while the testbench would be 600 to 800 lines. A production ready model will contain more ports, more robust assertions, more robust analog loads and drivers, timestep control and the smoothing of piecewise constant signals for signals that go into the circuit simulator. A production ready testbench will contain more ports, more test sequences, assertion tests, more complex sequencing, more complex waiting for settling behavior, more complex measurements, more complex comparisons, messages printed to the log file will be parsable, and tasks to encapsulate commonly used testbench activities.

D. Shortcomings of this Verification Flow

This flow will work to catch bugs in designs. The primary shortcoming of this flow is the cost or the amount of time needed to write the models and testbenches. The verification effort is typically 20% of the design effort [1]. For every 5 analog designers, 1 dedicated analog verification engineer is needed to write models and testbenches. For a team new to analog verification, the cost can be as high as 30% or 40% of the design effort. Plus, engineers with modeling skills are difficult to find, so even if one were willing to invest in this effort, one may not be able to hire the people to do so. Analog designers cannot solve the problem, because they often do not want to do it; they do not have the modeling and scripting skills; and they are overworked with what they need to do.

The other shortcoming is that there is an inherent inefficiency when the analog designer and the analog verification engineer work together. The analog designer needs to communicate to the analog verification engineer the information necessary to write the model and the testbench. Unfortunately, the analog designer typically only has an *ad hoc* specification written as a document. This specification is typically not up-to-date, is often ambiguous, and usually incomplete. Few look at the specification, and so writing the specification is of secondary concern. The schematic is the golden, so all priority is placed in the schematics. In fact, the schematic often serves as the documentation, where truth tables and descriptions are drawn onto the schematic. As a result, the verification engineer needs to look both at any paper specification that exists and at the schematic to understand what the block is supposed to do. The schematics are imperfect as a specification in that it is often difficult to discern from looking at interconnected transistors exactly what a complex topology is meant to do. As a result, much communication needs to occur between the designer and verification engineer. This is time consuming, error prone, and incomplete.

Because of the need for schematics, digital verification engineers cannot be recruited to help, because reading an analog schematic is usually not in the skill set of a digital verification engineer plus they are often not sufficiently familiar with analog specifications to be able to separate function from performance.

III. A Specification Driven Analog Block Level Flow

How do we address these shortcomings? First, we recognize that for all of the effort that is required, the analog verification problem is *not* a difficult one. Though many lines of code, analog functional models tend to be simple, because the underlying functionality of analog blocks tend to be simple. Analog is difficult to design, because it is difficult to achieve performance specifications, but the functionality of the block is typically simple. Functional tests are simple. Performance measurements can be complicated, and sometimes difficult to achieve in simulation, but functional tests tend to be straightforward. High-level languages such as Verilog, Verilog-AMS, and SystemVerilog exist for model writing and simulators that are robust exist.

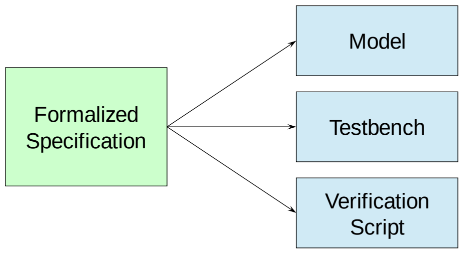


Figure 8. Using a formalized specification as a starting point

Given that the underlying functionality of analog blocks tend to be simple, one can write a formalized specification for analog blocks, one that is unambiguous and parsable. Complexity (logic and state) by design lie in the digital RTL. The specification can contain all of the information necessary to build the model, verify the circuit, and verify the model. This specification can be created by analog designers, by designing the “language” of the specification to be familiar and human readable. Once such a formalized specification exists, then a generator can be written to turn the formalized specification into a model for the analog block, and a generator can also be written to create a self checking testbench, one that validates that the analog schematics and the model generated are functionally equivalent. With the generator in place, this formalized specification is now an *executable* specification. This formalized specification serves as a starting point as shown in Figure 8 in the model creation process.

The advantages of using executable specifications are as follows. Using a human readable specification instead of a schematic or model facilitates communication. Any member of the design team can read the specification. No additional training is required. Facilitating communication reduces chances of error. Specifications are meant to be easy to read, understand, and update. Specifications are incredibly concise compared to models, and therefore, can be entered quickly. The executable aspect eliminates ambiguities, and makes it so that there is a reason for the specifications to exist and be kept up-to-date. Since the specifications contains all necessary information to describe the functionality of the block, it, therefore, can be used as the input to automatically generate the model and self checking testbench that validates that the model and schematic are functionality equivalent. Because the model is generated automatically in a correct-by-construction fashion, if the testbench runs successfully on the schematics, then we have shown that the *specification*, *model*, and *schematics* are functionality equivalent.

Using this approach allows models to be created early in the design process. This allows digital designers to created analog-dependent RTL sooner rather than later. The verified models are available to the digital designers and verification engineers, chip level verification engineers, system designers, test engineers, product engineers, end customer, and other analog designers to assist in analog design process.

Since the model and testbench creation time represents the bulk of the analog challenge, this automates a large percentage of the analog verification effort, and in return gives time for analog and digital verification teams to do a better job especially at the chip level. It allows the verification teams to focus on thinking about the verification itself such has how to achieve better coverage, and allows them to look at improved methodologies such as UVM, rather than the spend their time in the intricacies of how to properly to write efficient models.

IV. Example

Name	**Dir**	**Type**	**Description**	**Range**	**Behavior**	**Nominal**	**Instruments**
out	output	voltage	PGA output		V=en*Gain*V(in)		with tol=10m
in	input	voltage	PGA input			1.25	
[5:0] gain	input	signed	Gain (in half dB steps)		Gain=dB(gain/2)	32	
en	input	digital	Enable			1	
bias	input	ibias	Bias	5uA to 15uA		10u	
vdd	input	supply	Vdd	2.25V to 2.75V	I=100uA*en	2.5	flow: with tol=1m
gnd	input	ground	Ground	-10mV to 10mV		0	

Figure 9. Specification for a programmable gain amplifier

An example specification for a programmable gain amplifier (PGA) is shown in Figure 9. This specification is typical of one that a designer might use. It is in a table format, describing the pin names in the left most column followed by columns for the pin direction, type of pin, and a description for what the pin does. The data in the remaining columns are also typically present except for the right most one. The exact column names and format will differ between design groups. The range describes the valid input range of a pin. For example, the supply pin, “vdd,” is expected to receive a voltage between 2.25 volts and 2.75 volts. The behavior describes the input/output relationship for a particular pin. For the output pin, “out,” the voltage output is to be the enable pin, “en,” either a 1 or 0, multiplied by the variable, “Gain,” multiplied by the voltage on the input pin, “in.” The variable, “Gain,” is defined as the input pin, “gain,” converted to voltage gain using a dB conversion function. There are behaviors that cannot be easily described in equation form, e.g. state machines or a circuit topology. For those behaviors, other tables can be used. The nominal describes the default value for the inputs. For example, the default value for the supply pin, “vdd,” is 2.5 volts. The instruments column describes how well the model and schematic need to match. For example, when applying the same stimulus to the model and the schematic, the difference in voltage on the output pin, “out,” of the model compared to the voltage on output pin, “out,” of the schematic should be less than 10mV. The specification also indicates that the supply current on “vdd” needs to match by less than 1mA. The primary difference between the specification in Figure 9, and a typical datasheet, is that the specification in Figure 9 is specified in an unambiguous manner, making it machine interpretable. It is also a better practice to have no ambiguity in specifications to minimize errors in interpretation.

Although this example is simple, it is representative of a production PGA in the type of pins, just not in the number of pins. A production PGA would likely have more control words, perhaps multiple enable lines, multiple bias lines, and possibly more supply and ground lines. To specify such a PGA, it would just be a matter of adding more rows to this description.

The insight in this specification based approach is to recognize with just a very small amount of information, what is shown in this specification, that one can generate a pin accurate functional model either by hand or automatically. This specification is also sufficient to generate a testbench that validates that the model and schematic are equivalent. The testbench will go through all settings on “en” and “gain,” thus providing full functional coverage with respect to the digital inputs. The model and testbench can contain all of the complexities needed for a robust model as explained in Section IIC.

We developed a tool to translate a description such as the one shown in Figure 9 into a Verilog-AMS model. The Verilog-AMS model that corresponds to this specification is 196 lines of code, and the testbench is 663 lines of code. Writing this model and testbench would be tedious and time consuming, and would not add any additional value incrementally, since a computer program can make this translation automatically. Of course, there is value in the first time the model and testbench is written in how to write effective Verilog-AMS code. Compared to the specification, reading the model is a challenge. It is far easier for a designer to read a specification than to read a 196 line model. Even though the model and testbench can be designed to be well commented, easy to read, only approximately 20% [1] of analog designers are skilled at reading models, and probably far fewer are capable of looking at a model that includes the complexities required for making the model robust. Most analog designers should be able to understand what is being communicated in Figure 9.

Our tool provides a web based interface for the specification entry. The data can also be provided in JSON [5], an easy to read lightweight data interchange format, if one wants to write a script to generate specifications. JSON can be imported/exported by many scripting languages including Python [6], Perl [7], and Ruby [8].

We have had success with this approach, extending it beyond PGAs, and beyond models with only 12 pins. The approach has worked for most analog functional blocks, e.g. regulators, amplifiers, bias generators, data converters, etc; and covers virtually all leaf level cells. It has also been found to be scalable working fine with blocks of 100 to 200 pins. This specification can be extended to add performance, edge detection and state, and testbench control including measurements, sequence control, and tolerances.

V. CONCLUSIONS

With specification driven verification, analog designers can perform block level verification under the guidance of analog verification engineers, and create verified block-level models. The analog verification engineers focus on verifying the analog section in its entirety. These models would then be shipped to the chip-level verification team, who would then apply standard chip-level verification techniques with the analog.

With specification-driven analog verification, we can fully verify transistor level analog design in a manner that is practical, efficient, systematic, scalable, and repeatable. The manual approach of writing models with an analog verification methodology was already practical. This methodology is known and the effort was bounded to approximately 20% of the design effort. Automation with the specification driven approach only makes analog verification more practical. The specification driven approach makes the effort more efficient. It is systematic in that the same approach can be applied for most analog blocks and just needs to be repeated. It is scalable in that the amount of effort as the number of pins increase on a block goes up linearly or less. It is repeatable in that this approach is self checking. The verification effort can be repeated in an automated fashion.

Finally, this approach enables early co-design and co-verification of analog and digital, and enables true chip level verification which requires the inclusion of the analog content on a chip. The analog specifications or the schematics represent the "second pair of eyes" in analog verification.

ACKNOWLEDGMENTS

This flow is based on many verification efforts on many types of chips including power management, audio, data communications, RF, etc. in which we have been involved. We would like to thank all of our colleagues at the many different companies with whom we have worked.

REFERENCES

[1] Based on Designer's Guide Consulting, Inc. experience in verifying many analog/mixed-signal chips.

[2] H. Chang, K. Kundert, "Verification of Complex Analog and RF IC Designers", *The Proceedings of the IEEE*, February, 2007.

[3] H. Chang, K. Kundert, "Introduction to Analog Verification," *IEEE Solid-State Circuits Magazine*, Vol. 1, Issue 4, Fall 2009.

[4] http://www.designers-guide.com/newsletters/0711/index.html

[5] http://json.org/

[6] https://www.python.org/

[7] http://www.perl.org/

[8] https://www.ruby-lang.org/en/