Designer's Guide Consulting
Analog, Mixed-Signal & RF Verification

Analog Verification Newsletter

Number 3, September 2007

Contents

Greetings

Dear friends and colleagues,

This month's article is entitled "Modeling Analog for Verification is Easy." We have some new announcements and several reminders from last month. This month, we have added a new section, entitled, "Responses from Our Readers," and have included several. We appreciate all of the feedback!

Sincerely,

Henry Chang and *Ken Kundert*

Announcements

Ken to Speak This Sunday at CICC on Analog Verification

September 16 at the DoubleTree Hotel in San Jose, California.

See Ken Kundert talk on Analog Verification at the IEEE Custom Integrated Circuits Conference Educational Session on Sunday, 9/16.

For more information visit *www.designers-guide.com/partners/cicc4.html*.

Custom Integrated Circuits Conference

September 16-19 at the DoubleTree Hotel in San Jose, California.

Learn what's new in analog, mixed-signal, RF design at the IEEE Custom Integration Circuits Conference.

For more information visit *www.designers-guide.com/partners/cicc3.html*.

Behavioral Modeling and Simulation Conference

September 20-21 at the DoubleTree Hotel in San Jose, California.

The 2007 IEEE International Behavioral Modeling and Simulation Conference is a "workshop-like" conference that focuses on behavioral modeling and simulation for analog electronic circuits and systems. As such, it addresses the development and application of behavioral languages and simulators, as well modeling practices and the extraction of models. The Verilog-AMS and VHDL-AMS languages are of particular interest. The conference will held in conjunction with the

Custom Integrated Circuits Conference at the Doubletree Hotel in San Jose in 2007.

For more information visit *www.designers-guide.com/partners/bmas1.html*.

Henry to Speak on Analog Verification at BMAS

September 21 at the DoubleTree Hotel in San Jose, California.

Henry Chang will be giving the Friday keynote address at BMAS. He will be discussing analog verification.

For more information visit *www.designers-guide.com/partners/bmas2.html.*

Ken to Speak in Boston at CMRF on Analog Verification

October 3 at the Marriott Long Wharf in Boston, Massachusetts

Ken Kundert will be speaking on analog verification at CMRF in Boston. CMRF is a one-day workshop on compact modeling for RF & microwave applications.

For more information visit *www.designers-guide.com/partners/cmrf1.html*.

Modeling Analog for Verification is Easy

By Henry Chang and Ken Kundert

When we teach our class or assist an analog, mixed-signal, or RF design group with analog verification, we explain that the three most time consuming tasks in analog verification are modeling, regression test development, and debugging. This article will address the first of these tasks. Many engineers do not like to write analog models. They perceive model writing to be difficult and time consuming. They complain that the models never give enough detail and that keeping them consistent with the design is difficult. Our approach focuses on making analog modeling easy.

Following a few simple rules can avoid most of the pitfalls in analog modeling.

- Model only what is necessary
- Model at the highest level possible
- The model must be pin accurate
- Use model coding guidelines
- Hold model reviews

We will describe how to apply these concepts in modeling a 4 bit flash analog-to-digital converter (ADC) [maxim apnote 810] found in this generic analog front end (AFE) example shown in Figure 1. The input, “AFE In,” goes through a receive channel that does some analog signal processing such as filtering and then into the ADC. The 16 bit thermometer encoded output of the ADC goes to a digital core for further signal processing. To support the ADC, there is voltage regulator to supply power, a bias generator that feeds the ADC bias currents, and a clock generator that provides the clock. The digital core can control the flash ADC using a power down signal and by changing the gain at the input of the ADC.

The specifications for the ADC are shown in Figure 2. The block diagram and specifications comprise the key pieces of information needed to start modeling. Ideally, model writing starts early in the design process. Often at this stage, block diagrams and specifications have not yet been committed to paper. This should not a problem. Talk to the lead designer and block designers.

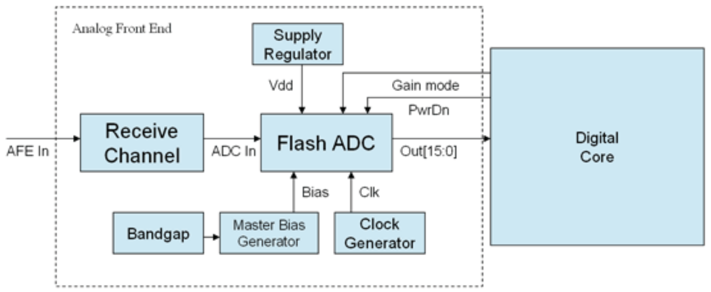
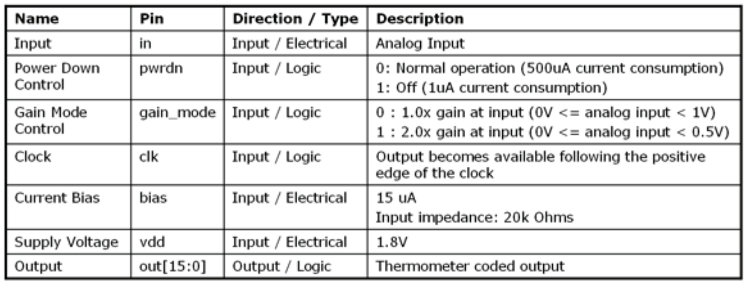


Figure 1: Analog front-end example.

Help them write down the block diagram and specifications. Developing a first draft of these can only help the entire team.



Name	Pin	Direction / Type	Description
Input	in	Input / Electrical	Analog Input
Power Down Control	pwrdn	Input / Logic	0: Normal operation (500uA current consumption) 1: Off (1uA current consumption)
Gain Mode Control	gain_mode	Input / Logic	0 : 1.0x gain at input (0V <= analog input < 1V) 1 : 2.0x gain at input (0V <= analog input < 0.5V)
Clock	clk	Input / Logic	Output becomes available following the positive edge of the clock
Current Bias	bias	Input / Electrical	15 uA Input impedance: 20k Ohms
Supply Voltage	vdd	Input / Electrical	1.8V
Output	out[15:0]	Output / Logic	Thermometer coded output

Figure 2: Analog-to-digital converter specifications.

We'll now describe our modeling rules in the context of this design. The first rule is "model only what is necessary" to fulfill the verification plan. The verification plan is the first step in analog verification. By the modeling stage, a first draft of the verification plan should be completed. Suppose in this case, what the verification plan says is that we want to use analog verification for what it does best, functional

verification. We want to check all of the inputs and outputs to the AFE and verify that the blocks functionally behave as expected. We decide that rather than model second order effects such as integral non-linearity (INL) in the ADC, we will count on mixed-level transistor simulation for this, and if it turns out that it is too slow, only then will we consider putting this effect in the model but with much trepidation. Even though adding this effect in the model might require adding only a small number of lines of code, avoid the temptation. The verification plan doesn't require it, and if we were to add INL effects to the model, the modeling effort now has all those issues that engineers complain about when writing analog models. The model is now difficult to write, inaccurate, and difficult to maintain. The modeling is difficult, because the ADC now has to be characterized to understand exactly what the nonlinear behavior is. It is inaccurate, because there are many contributors to making a flash ADC non-linear such as reference voltage mismatch and comparator offset. These effects are very difficult to model accurately, because much of it is due to process variation. Finally, it is difficult to maintain, because as soon as the design changes, the laborious task of modeling has to be re-done.

Our second rule, "model at the highest level possible" gives two primary benefits. First, the I/O characteristics tend to be better defined since the specifications are usually clearer at the higher levels. Second, a lot of the detailed analog signals can be abstracted away, and thus the model is simpler to write. The issue preventing us from modeling the entire analog section of the chip as one block is that simulating the entire analog section all at the transistor level is likely too slow, and we will be unable to validate that the model matches the implementation. So, the way we choose where to model is to start from the top level trying to find blocks that are mostly analog that have relatively clean and defined interfaces. Then we look or try to guess the transistor level implementation to see if it can be simulated in a reasonable amount of time. Even if the design has not been started which is often the case, there is usually a previous version of the block to get a feel for how long a simulation might take. We define reasonable simulation time as taking no more than few hours for a test set. In this ADC, a test set might be to power-up the ADC, put it into one of the gain modes, put in 16 voltage levels, and check the digital output. If at transistor level this simulation takes only a few hours, then this is where we should begin. If it takes longer, we'll need to model at a lower level in the hierarchy. If the simulation runs quickly and this is also true for the other analog blocks at the same level as the ADC, we might consider going up a level in hierarchy.

Figure 3 is the block diagram for the ADC. There is often a strong temptation to want to model the blocks at this level. Blocks at this level often seem simpler and easier to understand. In fact, the opposite is usually true. Our modeling rules say that we should not model at this level. The pitfalls are as follows. Instead of one model for the ADC, we now have four, assuming the register file is all digital. We also have many more analog signals to consider (at least 35 in this example). The voltage references to the comparators and the current bias references between the local bias generator and amplifier and comparators were not signals we had to consider before. It's easy to get mired in these details. Also, there are not likely to be specifications for each of the blocks. Extracting behavior from looking at transistor schematics is tedious and error prone. Modeling also cannot start until later in the design cycle, because one has to wait until this level of detail has been developed. Finally, the design is much more likely change at this level, thus making the models more difficult to maintain.

The goal is sweep as much under the carpet as possible by staying at a higher level. You'll know when you get caught at modeling at too low a level when you're trying to figure out exactly what detailed schematics do — current mirrors, voltage references, etc. Try not to model any of this.

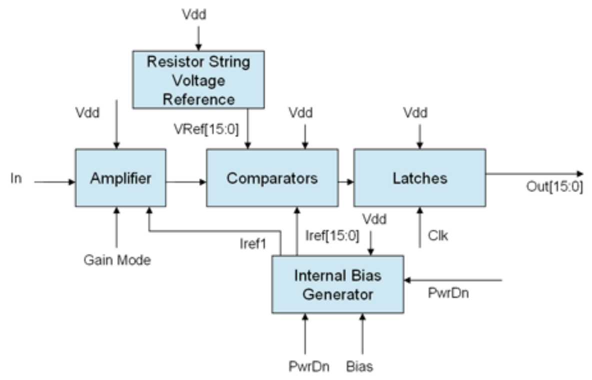


Figure 3: ADC block diagram.

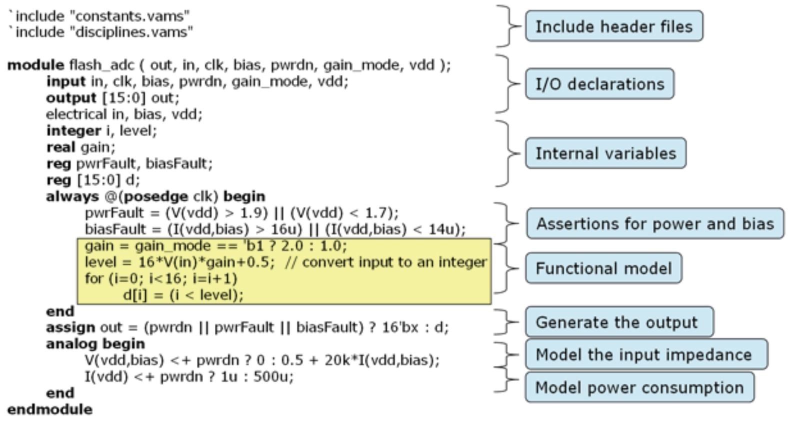
There are a couple of exceptions to this "model at the highest level possible" rule. The first is where at the block's top level, there is a lot of digital circuitry. We want to avoid re-modeling digital blocks and just use the digital block as is. If this is the case, we want to go one level down and apply this rule on the analog blocks remaining. Another exception is where the block is composed of many instances of only one or two analog components. In this case, it may be advantageous to write models for just the components, and let the block level schematic compose the overall function.

The third rule is that the model must be pin accurate. The pins on the models must exactly match the pins on the schematics. Buses, supplies, and the inputs and outputs must match exactly. Our assertion is that if the model is not pin accurate, it is not a true model of the implementation. Only with pin accuracy, can the model be verified to functionally match the implementation.

Our fourth rule, “use model coding guidelines,” simply instructs that one should follow some standard conventions when writing models. These might include internal signal naming conventions. They might be indenting and commenting style guidelines. It may include conventions on the order in which to place the code. For example, place the assertion checks first, then the function, and always put the analog section near the end. Largely, this is just a matter of having a common way of doing things. The benefit is that this will lead to models that are easier to re-use, easier to maintain, and understandable to others.

Finally, “hold model reviews.” Have the model writer walk through the models with other engineers. Even an expert modeler can forget certain issues. Model reviews can be very instructive to engineers who are not familiar with analog modeling. Finally, often there are many ways to model the same effect. Having model reviews can be helpful in choosing the best approach.

Based on these five rules, we have developed the model shown in Figure 4. The blue boxes show the major sections of our model. In yellow shows the code for the actual function. Because we modeled only what is necessary, the function is straightforward. Because we modeled at the highest level possible, we only have one model and it is a modest 24 lines. We started with including the standard Verilog-AMS header files. Using the coding standards, we place the I/O declarations immediately after the module declaration. We follow that with declaring our internal variables for which we follow a convention in naming our fault variables. We then place our assertions followed by the functional model. The assertions check that certain conditions such as Vdd and bias are correct for the operation of the ADC. Finally, we assign the output. We put the analog section at the end. This models the bias input impedance and the power consumption of this block.



```
`include "constants.vams"
`include "disciplines.vams"

module flash_adc ( out, in, clk, bias, pwrdn, gain_mode, vdd );
	input in, clk, bias, pwrdn, gain_mode, vdd;
	output [15:0] out;
	electrical in, bias, vdd;
	integer i, level;
	real gain;
	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);
		gain = gain_mode == 'b1 ? 2.0 : 1.0;
		level = 16*V(in)*gain+0.5;  // 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 4: ADC functional model written in Verilog-AMS.

Finally, note how most of the model is in the purely digital Verilog portion and will thus run in the event-driven engine. Modeling at this level will make these models run very fast.

How do we know this model matches the transistor level implementation? That's done as part of the rest of the analog verification process. In short, we create a regression test that walks through all of the inputs and modes of the ADC to see if we get what is in the specs. Then we re-run the same test on the transistor level implementation of the ADC. If the regression tests generates no errors on either, then the model and the transistor level implementation are consistent.

We've found that when applying these five rules, modeling can be a relatively straightforward task, and maintaining the models hasn't proved to be too difficult. Good luck.

Do you have a "rule" you'd like to offer? We'll publish them in our next newsletter.

Responses from Our Readers

First, we would like to thank everyone who wrote to us. It's great to get feedback and we're gratified to see that there is interest in analog verification. In fact, we received a couple of responses that require more thought and a longer response. We'll likely respond to those in future articles.

Specifications never capture everything; regression tests are time consuming to write

This reader responded that (1) specifications never capture everything, and that (2) in general, specifications are well communicated, but sometimes someone forgets to tell the modeling team. As a result, inconsistencies occur. He felt that our approach using regression testing would work, but asserts that this would require a lot of time and discipline and postulates that for every five designers there would need to be two full time verification engineers.

To point (2), we agree. Analog verification does require time and discipline. Our experience has been that for today's moderately complex design, for every five designers, there has to be one full time verification engineer. If you are applying analog verification for the first time, there may need to be 1.5 verification engineers. Further, as design complexity grows in terms of the number of modes, settings, and sequences, the verification effort increases. This is likely to be the case in the future. With this said, this does not mean that the overall number of engineers on a project grows when analog verification is added. One analog design manager who confirmed this 1 to 5 ratio explains that the verification engineer offloads work from the analog designer and is instrumental in catching architectural and implementation errors earlier. This time savings keeps the overall engineering effort for the analog design constant or lower, meaning if it takes 5 analog engineers to design a block, with analog verification fully in place, it still only takes 5 engineers, 4 dedicated to design and 1 to verification. And in the latter scenario, the overall engineering effort is further reduced because with analog verification, the time to find bugs in silicon and the number of re-spins is reduced.

To point (1), we have observed this as well. We find that when modeling, besides looking at the specifications, we need to spend time with the analog designers making sure that we are modeling everything that is needed. We use a checklist of questions to make sure that we are asking the right questions. We often add what we learn to the specifications. In this way, we are still consistent with our methodology where the regression tests are based on the expected results in the specifications. By sticking to the "modeling at the highest level possible" rule in our article, "Modeling Analog for Verification is Easy." we can usually get the specifications needed.

Can top-down design be applied to leading edge designs?

This reader asks whether or not top-down design is realistic on leading edge technologies. He states that when it comes to leading edge technologies, it is often not known what the architecture is up front, and thus the interfaces cannot be nailed down. A lot of time is spent building performance models, and by the time the architecture is finished, there is only time to build bottom-up functional models.

To respond to this, we'll begin by talking about analog verification, and then answer the question on top-down design. Our experience has been that analog verification can be applied on leading edge technologies. We do not expect that the architecture and all of the interfaces have been nailed down up front.

A typical scenario we see for a new process where there are many unknowns is that the analog designers start by building a test chip to test the architecture and components. Typically, this design does not include the digital sections, has more knobs and settings than will be in the final design, may contain experimental blocks, and have additional circuitry to emulate the environment of the final chip.

What we claim is that analog verification can begin on day 1 of the production chip design cycle. It is true that a lot of changes will occur between the test chip and the final chip, but with some of the conclusions reached from the test chip, there is usually enough information to begin the functional models and the regression tests.

Without knowing the details of the reader's situation, we give a scenario of why he might see the problem he describes. Suppose his team is designing the analog front end that is shown in Figure 1 of the "Modeling Analog for Verification is Easy" article. Even prior to the test chip, the design team knows that there will be an ADC, but after the test chip, they realize that the number of bits in the ADC needs to change from 4 to 6, and from this they realize that a flash architecture is not appropriate. If the modelers do not follow the "model at the highest level possible" rule in the article, they can exactly get into the situation this reader describes. For example, they decide to model the ADC at the level shown in Figure 3 of the article. The modelers are stuck, because at this level the interfaces and function of the models are highly dependent on the architecture. Suppose instead we decide to follow the rule and model the ADC as one model. And suppose, prior experience tells us that in all likelihood even with the new architectures we have in mind, this 6 bit ADC can still be simulated by a circuit simulator at the transistor level in a reasonable amount of time. Thus, we start modeling right away by building a model of a 6 bit ADC with the basic pin information that is available such as the input/output relationship, and that there will be a gain mode selection. We also implement the regression tests to test those pins. When the architecture, more modes and settings are known, we add to the model and regression test. For example, chances are by going away from a flash architecture, latency will be introduced in the ADC. Adding latency to the model at this level is trivial and we do so when the latency information is available. The goal is to have the model and regression tests ready by the time the analog designer finishes the first draft of the implementation. At this point, we can run the regression test in a mixed-level fashion on the implementation to verify that the design meets specifications in all of its modes and settings in the context of the analog front end.

This reader asks, "what did we do wrong?" Given this scenario, we suspect that they modeled at too low a level. Also, since it sounds like they have a modeling group doing both performance and functional modeling, it may be that they need additional resources to focus on functional modeling. To get the full benefit from analog verification, functional modeling also has to be done early in the design cycle. We invite this reader to let us know.

Finally, we'll answer the reader's question in regards to "top-down design." Our experience has been that most designers follow top-down design to some extent. They start with an architecture and build system level models, such as in Matlab. When they have a first cut at an architecture, they talk to the circuit designers and negotiate what is possible. The circuit designer may have to do some exploratory design work to see what can be done. The results are fed back up to the system design team. They then iterate to converge on a specification.

Analog verification is meant to fit into this type of design methodology. Further, analog verification can be a stepping stone to designing in a more "top-down" fashion, because functional models are built as part of analog verification. These can be used to bridge the gap between the system level models and the implementation. Further, analog verification usually brings the benefits of what one is seeking from top-down design, such as catching errors early, enabling re-use and capturing design intent, having system models before the design is done, smoothing out the design process, and making the design process more systematic and repeatable.

Where do you teach your classes?

We received several inquiries on wanting to hear our lectures. Currently, we only offer on-site training. We are considering web-based classes. Please let us know if you would be interested in taking a class in this format. If we get enough interest, we'll consider offering them.

Can I contribute to your newsletter?

Absolutely! We invite any article that helps the analog, mixed-signal, RF designer or verification engineer. The article should be vendor neutral and provide insight on how the designer or verification engineer can better do their jobs. We are considering creating a category of "sponsored" articles that can focus on a specific tool as long as they are technical in nature and benefit the community. These sponsored articles would be in addition to the vendor neutral articles we currently publish.