2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Model Validation for Mixed-Signal Verification

The Importance of Being Earnest about Modeling

Carsten Wegener, PMIC Analog, Dialog Semiconductor, Germering, Germany

***Abstract*—Modeling of analog circuits is an important pre-requisite for event-driven, mixed-signal, simulation-based, chip-level verification. While modeling techniques have received significant attention, many of these come without associated model validation techniques. If one is earnest and serious [1] about modeling, one should be concerned about validating the models used.**

In this contribution, we develop a model validation process and demonstrate this for a simple mixed-signal circuit: a digital IO buffer which can be viewed as a 1-bit DAC and ADC. For modeling, we chose the simplest technique available: pure digital signaling. We show that our model validation process is sufficiently comprehensive to yield high-quality models; model quality being measured by the number of bugs/issues in the model after we hand it over to chip-level verification.

We report results from applying this to a full chip ASIC with 50 digital IO pins of five different flavors. We give an example of bugs we have detected in the analog IO cell designs which are easy to overlook during analog block-level verification. Using digital modeling and applying our validation process, the example bug becomes highly visible due to X-propagation along the signal chain.

Keywords—mixed-signal circuit modeling and validation

I. Introduction

A Power Management Unit (PMU) is a mixed signal IC that controls and monitors the power consumption of peripherals in the SOC. Verification of a PMU is a challenging task, requiring a power-aware, mixed-signal, verification environment [2]. One of the challenges occurs when trying to approach it with real number modeling (RNM) methodologies [3]; this challenge needs to be addressed to benefit from simulations that are faster than with SPICE models.

Porting verification methods, like UVM, from complex digital design to analog-centric top-level design is a challenge, and in [4], the authors explicitly do not discuss the "question of verifying the functional equivalence of the model and the circuit." Before being concerned with model validation, the authors in [5] state that "high performance analog modeling is one of the challenging areas for Real Value Modeling usage: the models are not useful for detailed low-level modeling or very close to block level simulation."

While the above statements might not sound very encouraging, we decided to go even simpler than full RNM: we restrict modeling to pure digital, i.e. signals with values of 0, 1, z, and x. Correspondingly, we are modeling circuits whose basic behavior can be described with pure digital signaling, i.e. digital IO pads of a PMU.

In order to obtain high-quality models, we break down design complexity and "re-use sub-block designs and associated models," as suggested in [6]. Therefore, our analog block-design follows the guidelines for enabling netlisting of a top-level model:

- Take advantage of the netlist where possible - model at the smallest feasible analog sub-block level.
- Partition digital logic from analog functionality - even locally.
- Use signal-flow input to output – 'inouts' are to be avoided where possible.
- Encapsulate analog-feedback within the model where possible.

Following this design style, we enjoy the benefits: the top level model is simply netlisted; so easy to create, and the port names, directions, bus widths, etc. are automatically correct. The leaf cell behavioral models are simpler, easier to write, and not as error prone. The top level model is largely correct by construction. In [7], the authors state that "unexpected behavior can be found due to the interaction of the lower level models that often points to real analog problems," and we can confirm this observation.

Because of the tremendous effort going into modeling, model validation is a key topic. A state-of-the-art UVM environment gives little comfort if the underlying models of the analog blocks are not adequate. Let these be models of IO buffers, analog comparators or complete sub-systems like a buck converter. For example, in [8], stimulus is applied to both model and schematic with checkers used for output comparison as shown in Figure 1. However, in the example shown, analog cell supplies are excluded from the stimulus as the authors' focus was on (digital) functionality of the signal processing. Moreover, it is not established in [8] that the applied stimulus is yielding sufficient coverage and that all relevant input combinations are considered.

In contrast, we account for all pins of the modeled analog cell including supplies, and we use exhaustive all-pin stimulus. This way, our proposed model validation procedures ensure that the intended behavior is covered by the applied stimulus, and that, for all valid outputs of the event-driven (aka Real Number) model, the analog schematic yields the same output response.

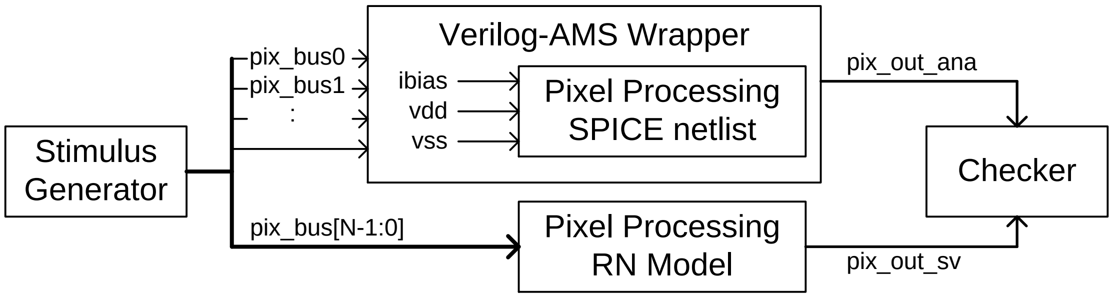


Figure 1: Testbench for verifying pixel-processing real number model [8].

II. IO Buffer Modeling

Standalone PMUs, or low-power SoCs, integrate a variety of multi-rail hard macros, like IO pads, which propagate the supply and signals from the analog to digital worlds. These IO pads are not actually part of the digital design. They are analog / mixed signal components and the digital supplies are not available to them [9]. Therefore, digital verification needs to include verifying the availability of the IO pad supplies when using them for signal transmission from digital to the analog chip environment.

A. *Circuit description*

A simplified schematic drawing of a dual-rail input and output buffer [10] is shown in Figure 2. The signal at the PAD travels from right-to-left and is digitally presented at dout when the input path is enabled (in_e=1).

For the output path, the digital signal din is driven left-to-right towards the PAD. When the output path is enabled (out_e=1), din=0 switches the PAD to ground by activating the NMOS gate NG=1. For din=1, the PMOS is activated by PG=0 only if the push-pull output mode is selected (pp_e=1). The pull-up and pull-down resistors are activated by the corresponding control signals pu_e and pd_e. The configurability of this cell allows for open-drain output functionality with either an internal or external pull-up.

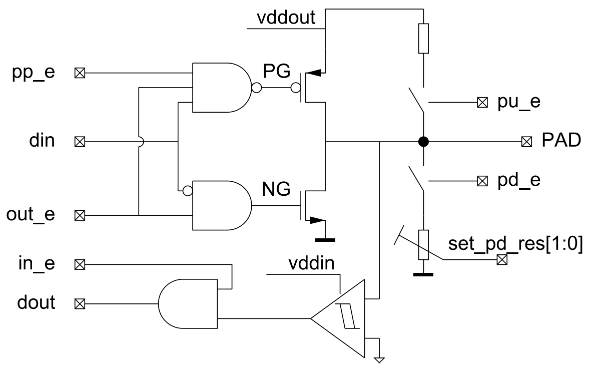
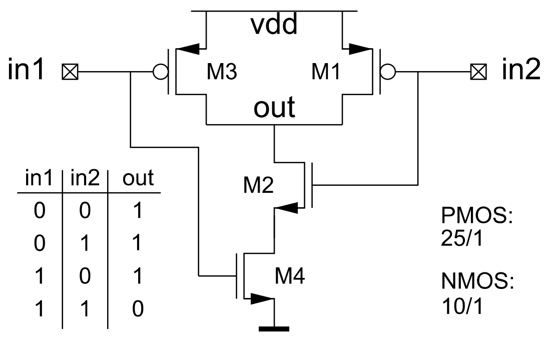


Figure 2: IO buffer simplified schematic.

B. *Circuit modeling*

Logic gates in Figure 2 are taken from the technology library. Using a 2-input NAND gate for the example, the schematic and logic table are shown in Figure 3, together with the corresponding Verilog model including supply checking.



in1	in2	out
0	0	1
0	1	1
1	0	1
1	1	0

```
module NAND2 (
 input gnd, //# TYPE=G
 input vdd, //# TYPE=P, HI=2.5
 input in1,
 input in2,
 output out //# TYPE=D, OK=DUT.supply_ok
 );
 assign supply_ok =
        gnd===1'b0 || (in1===1'b0||in2===1'b0)
     && vdd===1'b1 || ({in1,in2}===2'b11);
 assign out = (supply_ok) ? !(in1 && in2)
                          : 1'bx;
endmodule
```

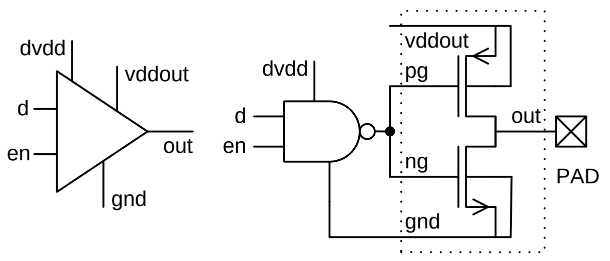
Figure 3: Logic table, transistor-level schematic, and Verilog model of 2-input NAND gate.

The basic digital operation of the NAND gate is only guaranteed for an adequate supply condition, indicated by supply_ok=1. In practice, the supply condition required is: vdd=1 and gnd=0. For illustration, we chose to model two exceptions:

1) both logic inputs in1 and in2 are logic-1, then the output out=0, despite vdd not being logic-1;
2) either of the inputs is logic-0, then the output out=1, despite gnd not being logic-0.

The importance of supply checking is not obvious for pure digital logic circuits. However, for circuits bridging the digital and analog domains, i.e. Digital-to-Analog and Analog-to-Digital Converters (DACs and ADCs), the relevance of the supply and reference voltages is obvious. A violation of the supply requirements is modeled by forcing the model's output to 1'bx. However, this information needs to propagate; thus, the RTL coding style employed in the output receiver needs to avoid X-optimism [11] for successful power verification.

Consider the 1-bit DAC in Figure 4 with the digital enable input en. The digital data d (referenced to supply dvdd) is converted to the output out with reference to vddout.



```
module cmos_out (
 input gnd, //# TYPE=G
 input vddout, //# TYPE=P, HI=1.8
 input pg,
 input ng,
 output out //# TYPE=D, OK=DUT.out_ok
 );
 assign supply_ok = gnd===1'b0 && vddout===1'b1;
 assign out = (supply_ok) ? 1'bz : 1'bx;
 assign out_ok = supply_ok && (ng ^ pg)===1'b0;
 pmos P0 (out, vddout, pg);
 nmos N0 (out, gnd, ng);
endmodule
```

Figure 4: One-bit Digital-to-Analog Converter with digital inputs for data (d), and enable (en) driving the analog output out ranging between the supply references vddout and gnd; Verilog model of CMOS output stage comprising PMOS/NMOS transistors.

This DAC can be implemented by a NAND2-gate driving a PMOS/NMOS output stage as shown. The cmos_out driver stage can be modeled by the Verilog code shown, again including supply checking.

As a result of having a model for the CMOS output driver and the NAND2 logic gate, the IO buffer shown earlier in Figure 2 can be derived. Effectively, the IO buffer can be seen as a 1-bit DAC and ADC connecting to the PAD. The switchable pull-up/down resistors can be modeled similarly to the output driver, only replacing the nmos and pmos primitives in the Verilog model with their resistive counterparts, rnmos and rpmos.

C. Event-driven, Mixed-signal, Design Simulation

Following the Netlist-based Verilog Verification Methodology [12], we obtain a Verilog netlist of the IO buffer design with models instantiated for the logic gates, the push-pull output driver stage, and the resistive pull-up/down circuits. We can simulate the PAD behavior using an off-the-shelf digital simulator.

Typically, IO buffers on Power Management ICs [13] are more complex than shown in Figure 2. For example, the user can choose from different output supply references, e.g. vddout1-4, select the digital input din from a number of on-chip digital sources, etc. This configurability is provided by on-chip registers in the digital core, labeled as “Power Manager” in Figure 5. Registers are accessible via I^{2}C or other standard digital communication interfaces, in the block labeled “Multiplexed IO Extender.”

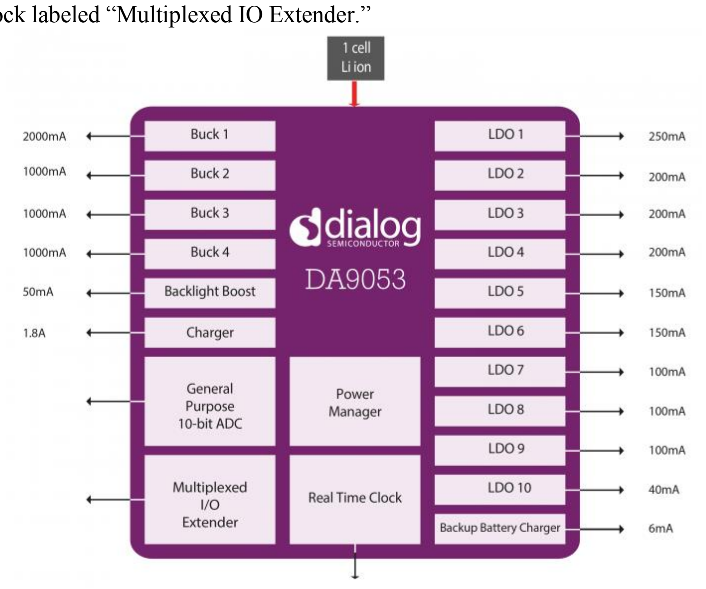


Figure 5: Block diagram of DA9053 PMIC with 5 buck/boost converters, 10 Low-Drop-Out regulators and two chargers.

A mixed-signal design simulation should allow the complete path to be verified, from the I^{2}C interface pins, to the digital registers, and to the PAD behavior. This simulation should also be sensitive to the required operating conditions, e.g. that the selected vddout must be available, in particular, if this supply is generated on-chip and could be turned off with the control of another register in the Power Manager block. The problem resulting from this configurability is that the verification effort for ICs increases exponentially with the number of managed power modes [14].

In our division, we are in the same position as described in [15]: the main features of our products are analog, thus, chip-level design is done by analog schematics. Without Verilog models for the IO buffer, we need to employ an analog simulator in order to translate the digital control signals to electrical behavior at the PADs of the chip. Full-chip simulations with all non-digital blocks being simulated at transistor-level is a huge effort in terms of engineering resource, license cost, and compute power. An event-driven simulation model is therefore attractive, especially for cases where the IO buffer sees high switching activities at its digital and analog interfaces.

Running full-chip simulations using Verilog models for mixed-signal circuits is attractive beyond IO buffer cells. In [16], we have demonstrated the feasibility of event-driven, mixed-signal simulations for a switched-capacitor-array-based, Successive Approximation Register (SAR) ADC, using Verilog. SystemVerilog with Real-Number-Modeling (RNM) has been explored in [17] as applied to buck converters for the purposes of test procedure verification.

III. MODEL VALIDATION AGAINST EXPECTATIONS

A prerequisite for model-based design verification is the availability of an “implementation model.” Such a model can be extracted from the design database by netlisting the connectivity of instantiated sub-blocks. Just as for digital circuits, we require an analog design style which composes an analog block of basic building blocks. Each building block has an associated model.

For this approach to be successful, it is mandatory to validate hierarchically: first the models at the building block-level and then the integration at the higher levels of the design.

A. Model types: specification and implementation models

For analog circuits, we distinguish between implementation models and macro- or specification models. The purpose of an implementation model is to represent the circuit implementation as is, including potential bugs to be detected by verification. This comes at the price of model complexity, as the model inherits (on purpose) all the connectivity and hierarchy from the analog block design.

In contrast, the purpose of a macro-/specification model is to represent the basic circuit functionality or specifications at the highest possible level of abstraction. Such macro-model can be completely independent of the implementation. However, the problem of validating the implementation versus the macro-/specification model is a non-trivial task which is beyond the scope of this contribution.

As an illustrating example, consider again the 2-input NAND gate. Let the Verilog code in Figure 3 be the implementation model, and the SPICE netlist in Figure 6 be the (transistor) implementation. The Verilog code shown in Figure 6 is now the corresponding specification model. While the SPICE implementation represents the four-transistor schematic (and is, strictly speaking, itself a model for simulation), the specification (given as a logic table in Figure 3) is expressed as an executable specification in Verilog syntax in Figure 6.

```
.SUBCKT NAND2 in1 in2 out VDD
M1 out in2 Vdd Vdd p1 W=7.5u L=0.35u
M2 net.1 in2 0 0    n1 W=3u   L=0.35u
M3 out in1 Vdd Vdd p1 W=7.5u L=0.35u
M4 out in1 net.1 0 n1 W=3u   L=0.35u
.ENDS NAND2
* use BSIM3 model with default parameters
.model n1 nmos level=49 version=3.3.0
.model p1 pmos level=49 version=3.3.0
```

```
module NAND2 (
 input gnd, //# TYPE=G
 input vdd, //# TYPE=P, HI=2.5
 input in1,
 input in2,
 output out //# TYPE=D, OK=DUT.supply_ok
 );
 assign supply_ok = gnd===1'b0 && vdd===1'b1;
 assign out = (supply_ok) ? !(in1 && in2)
                          : 1'bx;
endmodule
```

Figure 6: NAND2 SPICE implementation and executable specification as Verilog module.

The specification and implementation (Verilog) models are pin-compatible, such that they can be swapped for each other in an event-driven simulation environment. The aforementioned properties of the specification model, i.e. highest possible level of abstraction and implementation independence, are still fulfilled.

Due to the simplicity of the 2-input NAND gate example, the level of abstraction for both implementation and specification models is the highest possible. This is already different for the example of the 1-bit DAC shown in Figure 4. The DAC can either be modeled as composed of a NAND2 gate and the CMOS output stage, or it can be described in Verilog as a single module without sub-hierarchy. We leave the coding exercise of the specification model for the 1-bit DAC to the interested reader who might want to start with the NAND2 model in Figure 6, rename the module (NAND2→DAC1b), the logic inputs (in1→d, in2→en), and change the logic operation to out = d && en given the correct condition for the supplies dvdd and vddout.

B. *Formulating expectations*

Let us consider the example of the four-transistor NAND2 gate shown in Figure 3; the expected behavior is described by the logic table in the same figure. We need to demonstrate that the executable specification (given as a Verilog module in Figure 6) represents the specified logic behavior. Once this is shown, the executable specification and the implementation model can be compared in order to demonstrate/prove equivalence.

A straightforward check for the logic table being mapped to the specification model is to perform a transient simulation, apply all tabled input combinations, and check the expected output conditions. The Verilog stimulus and expectation is shown in Figure 7 with the waveform indicating that always stim_pass=1, and the output out=1'bx when supply_ok=0.

```
@(posedge stim_clk) {gnd, vdd, in1, in2} = 0;
@(negedge stim_clk) stim_pass = out===1'bx;

@(posedge stim_clk) vdd = 1;
@(negedge stim_clk) stim_pass = out===1'b1;

@(posedge stim_clk) {in1, in2} = 2'b01;
@(negedge stim_clk) stim_pass = out===1'b1;
@(posedge stim_clk) {in1, in2} = 2'b10;
@(negedge stim_clk) stim_pass = out===1'b1;
@(posedge stim_clk) {in1, in2} = 2'b11;
@(negedge stim_clk) stim_pass = out===1'b0;

@(posedge stim_clk) gnd = 1;
@(negedge stim_clk) stim_pass = out===1'bx;
```

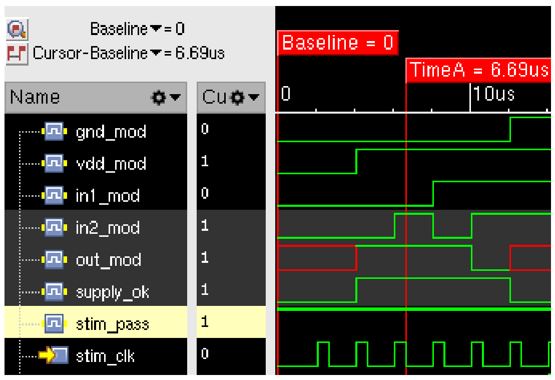


Figure 7: Verilog code (left) and waveform (right) for verifying NAND2 logic table from Figure 3 in a transient simulation.

With this approach, we have confirmed the logic table in Figure 3, but we have not yet exercised all input combinations. For example, in1=1'bx and in2=0 is missing from the stimulus. We could add this case to the stimulus description in Figure 7, but this approach does not scale for larger models, i.e. more inputs and outputs.

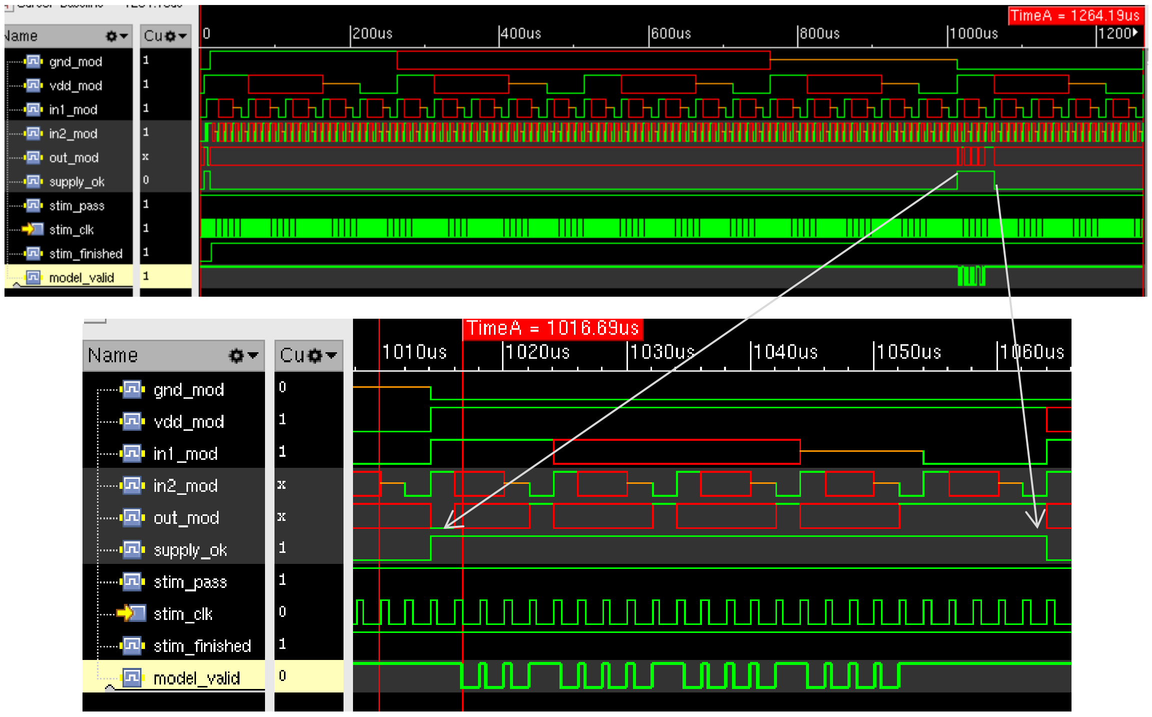


Figure 8: Simulated waveform with model_valid=0 for out=1'bx when supply_ok=1: full simulation time span (top) and zoom (bottom).

C. *Checking for X on outputs*

Instead of extending the hand-written stimulus in Figure 7, we apply 1'b0, 1'bx, 1'bz, and 1'b1 to **all** inputs in **any** combination, as shown in the waveform in Figure 8. Note that the 10µs simulation time for the hand-written stimulus ends with the rising edge of the signal stim_finished. After this point, stim_pass is always 1 as it has lost its meaning. Instead, the only criterion left is model_valid=1 indicating that out==1'bx if, and only if, supply_ok=0.

The actual condition for model_valid signal is determined by the output port declaration in the model itself:

```
output out   //# TYPE=D, OK=DUT.supply_ok
```

The //# character sequence starts an information string used to customize the test bench generated for the model. In particular, it indicates that this output is OK, i.e. 0, 1, or z, but not x, when supply_ok=1.

As apparent from Figure 8, there are input conditions leading to model_valid=0. These cases need investigation and lead us to introduce a new signal out_ok in the model. This signal is used to modify the out port declaration to OK=DUT.out_ok. Analyzing the root-cause for model_valid=0 leads to the following model revision:

```
    output out //# TYPE=D, OK=DUT.out_ok
);assign supply_ok = gnd===1'b0 && vdd===1'b1;
    assign out_ok = supply_ok && (!(in1!==1'b0 && in2!==1'b0) || {in1,in2}===2'b11);
    assign out = (supply_ok) ? !(in1 && in2) : 1'bx;
```

Formulating the conditions for out_ok can become lengthy. However, the approach is always the same: start with out_ok=supply_ok; simulate in order to find model_valid=0, and then either add valid exceptions or fix the model logic until you can sign-off on model_valid=1.

D. *Implementation model validation*

We apply the hand-written stimulus shown in Figure 7 to the implementation model in Figure 3. Unexpectedly, we find that the output out=1 when all inputs and supplies are zero leading to stim_pass=0. This example indicates the value of the hand-written stimulus: ease of debugging. The problem in our case is that supply_ok=1 despite vdd=0 because of a model coding mistake[1]. This mistake can be rectified by adding brackets:

```
   assign supply_ok =
       ( gnd===1'b0 || (in1===1'b0||in2===1'b0) )
     && ( vdd===1'b1 || ({in1,in2}===2'b11) );
```

Of course, the signal out_ok needs to be introduced and used in the implementation as previously for the specification model. With this last modification of the implementation model we can sign-off since model_valid=1 for all input combinations including stimuli with 1'bx and 1'bz.

[1] Coding mistakes are unavoidable; the important feature of model validation is a high chance of detecting such mistakes before the model is integrated in full-chip simulation where debugging becomes cumbersome and frustrating.

IV. Validation of Transistor-Level Implementation

A. *Implementation model versus specification validation*

For all we know, both the implementation and specification models fulfill the logic table and generate an output value 1'bx for the same condition of out_ok=1. However, the specification and implementation models differ in the definition of the supply_ok signal.

Generally, the supply_ok condition for the implementation model has to be a superset of the condition required by the specification model. This is obviously fulfilled by inspection of the Verilog coding shown in the previous two subsections. However, in general, this might not be so obvious, and moreover, the non-X output logic behavior might differ between the specification and the implementation models. Therefore, we need to simulate both the implementation and specification models side-by-side.

If both models yield non-X outputs, they must agree. The implementation model is **not allowed** to output X if the specification model behavior is non-X. In contrast, the specification model is **allowed** to output X if the implementation model behavior is non-X. This relationship is similar to the one between RTL and synthesized digital circuits: if an output in RTL is X, the synthesized circuit is allowed (even desired) to generate a defined non-X output, but the reverse is forbidden.

B. *Transistor-level implementation versus model validation*

Similarly to the setup in the previous section, comparing the transistor-level implementation with a model requires the test bench setup shown in Figure 9. The stimulus generator drives the model inputs with 1, 0, z, or x. The output(s) of the model are compared against the expectations by the "Spec checker" generating the signals stim_pass and model_valid. This (digital) part of the test bench we have already used to run the simulations described in Section III. These simulations run standalone on a digital simulator and do not need an analog simulator.

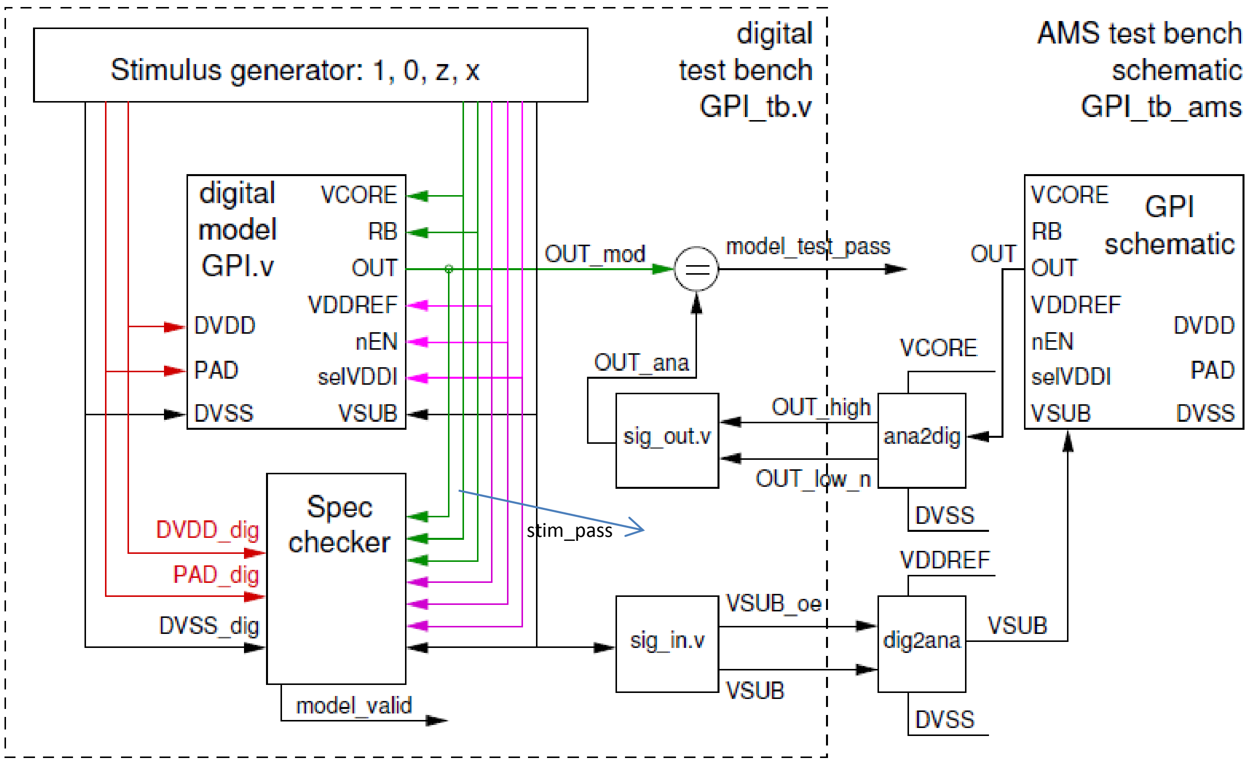


Figure 9: Digital stimulus generator and model embedded in AMS test bench instantiating the transistor-level schematic implementation.

In addition to the digital test bench, the surrounding AMS test bench takes the generated stimulus and provides it to the implementation schematic as voltages. The voltage level for logic-0 or 1 is by default 0.0 or 5.0V, respectively. However, this can be adjusted, e.g. logic-1→2.5V, by providing a parameter `HI=2.5` as in the vdd port declaration in Figure 3. By convention, the last declared power supply port (`TYPE=P`) determines the default logic-1 voltage level for digital ports. All ports are considered digital unless indicated otherwise. Thus, the NAND2 gate digital stimulus is straightforward to translate to analog voltage stimulus. The voltage response at the output out of the implementation schematic is translated to the digital domain as out_ana. This digital representation is checked for equality with respect to the model output out_mod. Equality is indicated by the signal model_test_pass=1. If the model output out_mod=1'bx, model_test_pass=1 independent of the implementation schematic response out_ana.

The model sign-off requirement is that model_test_pass=1 for all simulation time and all applied stimulus combinations. This indicates that the model responding with 1, 0, z predicts the implementation schematic response simulated for the same stimulus. A model response of X is a joker indicating that either the implementation response is undefined, badly conditioned, or one of 1, 0, or z.

In practice, and as shown in Figure 8, most input stimuli lead to undefined outputs because of invalid supply or input conditions. Therefore, we do not apply such stimulus to the implementation schematic. Thereby, we significantly reduce simulation time and avoid the analog solver to fail in finding a stable transient solution. With this in mind, it becomes apparent that checking the validity of the model outputting an X (as described in Section III.C.) is of utmost importance prior to model validation against the schematic.

C. *Verification results for IO buffer examples*

Let's consider a typical PMIC, as shown in Figure 5, and the requirement for pure inputs, open-drain outputs, and IOs with three selectable positive supply rails. Corresponding schematics are shown in Figure 10. The design task is to use the IO buffer from Figure 2 as a design block (with associated validated digital model) instantiated in the chip design with adequate digital control signal decoding.

For implementing the pure input shown, the IO buffer from Figure 2 is straightforward to use: tie unused signal inputs to logic-0 or -1, and connect the digital power-on-reset signal por_n to the in_e input of the IO buffer. Given this simple procedure, we are unlikely to make a mistake, and moreover, the digital model for the pure input is readily available by netlisting the design and can immediately be used for chip-level digital verification.

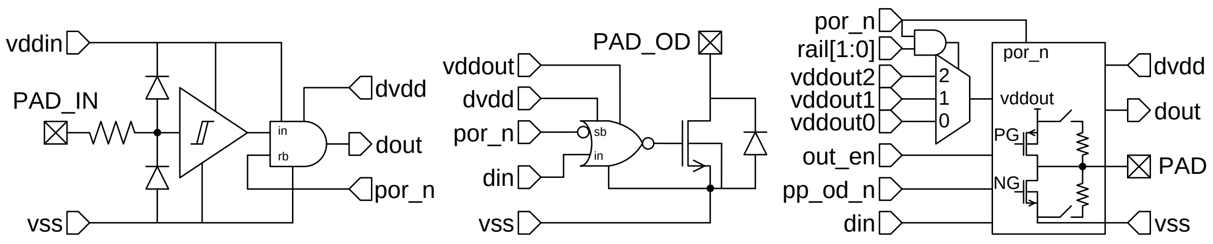


Figure 10: Simplified schematics for input-only (left), open-drain output (center), and input/output with selectable rail (right).

Still, as advocated in this contribution, we performed block-level checking of the new model against the expectation of the input cell functionality. As a result we observe that stim_pass=0 for almost all test cases. If we would have handed-over this model to digital chip-level verification, our "customers" would have blamed the model for X's being propagated to the digital core for no good reason. Moreover, debugging the model would have been very hard because of the "unreadable" gate-level-type of model formulation. Our chip-level verification community would have had arguments to go back to the previous approach: take the simple schematic in Figure 10 and formulate the model as follows:

```
    assign dout = por_n && PAD_IN;
```

without the need for complex hierarchy and X-generating supply-checks.

However, the actual situation is different: the model (of the instantiated IO buffer) reports an X on dout and PAD_IN because one of the supply checks in the model fails. The reason for this fail can be easily investigated in our block-level test bench: Not only the logic inputs, pu_e, pd_e, pp_e, out_e, set_pd_res[1:0], but also vddout is tied-low. For vddout=0.0V, the PAD cannot rise more than a diode voltage above ground without causing significant current to be sunk into the PAD pin. The issue is caused by the PMOS in the output driver shown in Figure 2, which is not apparent from the schematic in Figure 10.

As a result, we have detected a bug in the analog block-level design by applying digital modeling. This is great news; both for supporting our modeling approach and for the product development team as a bug was identified. In order to understand that this bug is hard to find in an analog simulation, consider the following: when stimulating the PAD_IN with a low or high voltage, in analog simulation, this voltage source supplies any current necessary, while no excessive current is taken from the circuit supplies. This highlights the importance of measuring ground currents during block-level verification.

D. *Model-based design procedure*

In the previous section, we designed the input-only buffer by instantiating the general-purpose IO buffer from Figure 2 with unused logic inputs and supplies tied to fixed levels. The same can obviously be applied in order to design the open-drain output shown in Figure 10, because this circuit also has a subset of the functionality implemented by the general IO buffer in Figure 2.

For the IO buffer with three selectable supplies as shown in Figure 10, the situation is different as we need to add new functionality: multiplexing the three supply voltages to the single supply input of the general IO buffer. This can be achieved in (at least) three ways:

1) Implementing the mux as shown in Figure 10 at the expense of adding three-times the silicon area for the PMOS used in the output driver stage. However, with this (digital-like) design approach, the series of mux and output driver PMOS shown in Figure 2 will never achieve the (analog) drive capability of the original IO buffer.
2) Instantiating the single-supply IO buffer three times, once for each vddoutX, and thus, driving the PAD from only one of the instantiations at any time. However, this design (re-use) approach, also instantiates three NMOS drivers, only one of which is being used.
3) Creating a new version of the CMOS output stage shown in Figure 4; now with one NMOS and three PMOS, i.e. one PMOS per supply input vddout0-2.

All options are functionally equivalent. Option 3) is optimal from an area vs. performance point of view, while option 2) is less area efficient, and option 1) needs more area and provides less performance.

From a modeling point of view, option 1) is the most straightforward to implement and shall serve as a specification model. During the (more complex) design of option 3) we derive an implementation model, and thus, we can verify the implementation versus the specification model.

V. Conclusions

In this contribution, we detail our approach to modeling and model validation as applied to IO buffers used on custom ASIC designs. By choice we use a very simple modeling technique: pure digital signaling but go into great detail when mapping the implemented circuit structure to the model of the functionality.

For model validation, we start with comparing the model against expected functionality in order to find basic logic coding mistakes that occur during modeling. We emphasize the importance of supply-checking and the validation of X's being generated by a model. The final validation step versus the transistor-level schematic is to ensure that non-X model behavior is matched by the design implementation.

This validation procedure can be applied to both specification and implementation models during the design development process. By example, we show how simple it is to introduce a design bug, e.g. during the derivation of the simple input buffer from the more complex IO circuit. Modeling the implemented structure and validating it against one's expectations is key to efficient design. The example bug just happens to be hard-to-detect with analog verification, unless you know exactly what you are looking for: unexpected ground currents.

Finally, we give an example of combining modeling and design, by first deriving a specification model which can be used as a functional reference for the implementation.

Acknowledgment

The author would like to thank Nir Dahan for requesting "a **fast** and most important **accurate** method of verifying the GPIOs in the system" which has led to the application of modeling in our product development flow and the development of the described validation methods.

References

[1] Oscar Wilde, "The Importance of Being Earnest, A Trivial Comedy for Serious People," Play first staged in London, 1895.

[2] A. Pandey, M. Welponer, and G. Kowalczyk, "Power-Aware Verification in Mixed-Signal Simulation," *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-8, 2014.

[3] D. Spigel and M. Hershcovitch, "With great power comes great responsibility: a method to verify PMICs using UVM-MS," *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-8, 2014.

[4] F. Assmann, A. Strobel, and H. Zander, "A concept for expanding a UVM testbench to the analog-centric toplevel," *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-6, 2015.

[5] D. Juneja, S. Prabhu, and S. Veluri, "Practical considerations for real valued modeling of high performance analog systems," *Proc. DVCON US*, Design and Verification Conference and Exhibition, pp. 1-10, 2016.

[6] J. McGrath, P. Lynch, and A. Boumaalif, "Comprehensive AMS verification using Octave, real number modelling and UVM," *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-8, 2015.

[7] V. Ranga, M. Subramanya, A. Shirwal, P. Rajan, and J. Goswick, "Unique verification case studies of low power mixed signal chips," *Proc. DVCON US*, Design and Verification Conference and Exhibition, pp. 1-13, 2016.

[8] K. Buescher, M. Becvar, G. Tumbush, and D. Jenkins, "Verification of an image processing mixed-signal ASIC," *Proc. DVCON US*, Design and Verification Conference and Exhibition, pp. 1-11, 2016.

[9] M. Jain, A. Singh, J. Bharath, A. Srivastava, and B. Jain, "Power aware models: overcoming barriers in power aware simulation," *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-13, 2014.

[10] C.Wegener, "ATPG for mixed-signal circuits using commercial digital tools," *Proc. IMS3TW, Int. Mixed-Signal, Sensors and Systems Test Workshop*, pp. 1-6, Sept. 2014.

[11] A. Naval and G. Jain, "Complex low power verification challenges in NextGen SoCs: taming the beast!" *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-9, 2014.

[12] J. Qui and K. Schwartz, "NVVM: a netlist-based Verilog verification methodology for mixed-signal design," *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-7, 2014.

[13] Dialog Semiconductor, "DA9053 flexible system level PMIC with multicore support." [Online]. Available: http://www.dialogsemiconductor.com/products/power-management/da9053, 2012.

[14] H. Bhatt and M. Prashanth, "Combining static and dynamic low power verification for the power-aware SoC sign-off," *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-6, 2014.

[15] A. Freitas and R. Santonja, "UVM ready: transitioning mixed-signal verification environments to Universal Verification Methodology," *Proc. DVCON Europe*, Design and Verification Conference and Exhibition, pp. 1-8, 2014.

[16] C. Wegener, "Method of modeling analog circuits in Verilog for mixed-signal design simulations," *Proc. ECCTD, European Conf. on Circuit Theory and Design*, pp. 1-5, Sept. 2013.

[17] E. Shera and C. Wegener, "Buck converter modeling in SystemVerilog for verification and Virtual Test applications," *Proc. IMSTW, Int. Mixed-Signal Testing Workshop*, pp.1-6, June 2015.