Predicting the Correlation between Analog Behavioral Models and SPICE Circuits for robust SoC Verification

Vipin Sharma* vipin@ti.com
Guha Lakshmanan* glakshma@ti.com
Sandeep Tare* tare@ti.com
Sudhind Dhamankar* sudhind@ti.com

*Texas Instruments Incorporated

ABSTRACT

Behavioral modeling of analog circuits is widely advocated and accepted means to speedup mixed-signal SoC level simulations. The validity of these simulation results is contingent on the quality of behavioral models used. This paper presents a verification methodology to establish equivalence of analog behavioral model and the SPICE circuit being modeled. The proposed methodology employs SPICE-on-top co-simulation environment to simulate the behavioral model in the same SPICE testbench that is used for circuit characterization using SPICE simulations. Circuit characteristics/metrics of interest are defined and checkers are developed to measure them. Results from SPICE and co-simulation runs are compared using automated checkers in a regression environment. A dashboard is generated showing PASS/FAIL status for every metric giving an accurate measure of correlation between the SPICE circuit and the behavioral model. These models are leveraged to run thousands of simulations to predict system performance which is not possible using SPICE circuits. This methodology is currently being used at Texas Instruments, Inc. to accurately predict system performance of complex mixed signal products.

1. INTRODUCTION

The verification methodology used in complex RF SoCs [1] like those used in single-chip GSM phones almost always requires development of analog behavioral models. There are several methodologies [2] and [3] that discuss different approaches towards behavioral modeling of analog circuits. All these methodologies address the computation challenge posed by the need to run large number of extremely long SoC level simulations. These methodologies allow prediction of critical analog characteristics of the system, such as phase noise, signal to noise ratio, DC offsets, process variations etc. From the simulation results, it may be inferred that the system functionality and performance matches very closely with the specification. However, there is uncertainty associated with the predictions of analog characteristics because these are based on results from behavioral models, not actual circuits. Some of these uncertainties could be because of incomplete modeling of analog behavior like sensitivity to bias currents. Some could be because of inaccurately modeling effects like circuit performance changes over various process corners.

One of the recommended methods to address this challenge is to co-simulate the SPICE circuit with rest of the system [4] and [5]. Commercial co-simulation tools impose several interface restrictions when discrete time (real number) behavioral models are used. Also, the computation overhead of co-simulating SPICE circuits at the SoC level makes this an unviable proposition. Another approach [6] talks about using calibrated models with generic parameters to account for circuit/process specific behavior. But the sheer number of specifications that the analog circuit must meet makes its reuse (and therefore reuse of the behavioral model) virtually impossible. Other approaches [7] and [8] discuss the use of the same stimulus for model and circuit simulations but do not give much thought to automatic results validation. This is the most common approach for behavioral model validation and is very error-prone because it relies on manual comparison of simulation results.

In this paper, a methodology is proposed to compare key analog characteristics of the behavioral model with the circuit. The idea is to restrict all SPICE and co-simulation runs at the block level, where the complexity is manageable and the simulation time acceptable. Emphasis is placed on defining the critical analog inputs/outputs and characteristics of each block and capturing these in checklists which are used as reference for model development and validation. Inputs for the checklists are gathered not only from the block/module specification but also from the circuit designer. The simulation environment (testbenches, checkers, simulator setup files etc) is built in such a way that every input combination of interest is exercised and every output metric of significance is measured. Each input/output combination constitutes a testcase. Each testcase is run using the SPICE circuit and the behavioral model and the degree of correlation is reported. If the behavioral model is found to be incomplete or inaccurate, it is enhanced and the system performance affected by this metric is re-evaluated. Although iterative, this process of identifying and fixing defects at the block level to improve performance at the system level is infinitely faster than attempting to run circuit simulations at the system level.

The paper is organized as follows. Section 2 introduces the BMV flow. Section 3 validates the flow with a real world testcase followed by Section 4 which explains the production use of the flow. Section 5 concludes the paper.

2. BMV FLOW

The behavioral model validation (BMV) methodology presented in this paper employs a SPICE-on-top co-simulation environment to simulate the behavioral model in the same SPICE testbench that is used for circuit characterization using SPICE simulations. Circuit characteristics or metrics of interest are defined and checkers are developed to measure them. Results from SPICE and co-simulation runs are compared using automated checkers in a regression environment. A dashboard is generated showing PASS/FAIL status for every metric. The entire flow is captured in Figure 1. Following sub-sections describe all the steps involved in the flow in detail.

2.1. BMV Flow Pre-requisites

2.1.1. Checkers

Automated checkers are important constituents of the proposed BMV flow. These checkers process the simulation outputs to calculate metrics (circuit characteristics) of interest for the circuit under test. For example, an important metric for a voltage amplifier would be the gain at the output. A typical checker for this circuit will process SPICE and co-simulation outputs and calculate the gain of the amplifier. Another checker can process the same simulation results to calculate another parameter of interest such as output noise. Checkers are specific to the circuit under test (CUT). Therefore, every new circuit or metric requires the development of a new checker. Section 3.2 explains the checker development process for the DCO (Digitally Controlled Oscillator) circuit which was used as a case study. Following sub-sections present other requirements for the BMV flow which can proceed simultaneously with the development of checkers.

2.1.2. Analog Circuit and Model Consistency

Another flow prerequisite is the consistency of module and port names between behavioral model and analog circuit. This is necessary because the SPICE testbench used to simulate the analog circuit is also used to simulate the behavioral model during co-simulation runs. Having the same SPICE testbench for both co-simulation runs and SPICE runs allows the user to keep a consistent interface between the blocks and the SPICE testbench.

2.1.3. Parameterized Test Benches

The use of parameterized SPICE testbenches provides the flexibility to configure circuit inputs such that all the functional and test modes of interest are exercised. For a simple voltage amplifier whose gain is controlled by two digital inputs (thus giving four different gain settings), four different modes of operation are possible corresponding to four different gain settings. This necessitates four different test cases for this block. The SPICE test bench for each test case must be configured such that these two digital inputs are varied to exercise all possible gain settings. This configuration is only achievable if the digital inputs are defined as parameters in the test bench. For every test case, the parameter values are generated dynamically in a separate file called "parameter file" that is included from the SPICE netlist. The four gain voltage amplifier, thus, needs four different parameter files. Each file configures the circuit under test in one of the four gain modes.

2.2. BMV Flow Setup

With all the pre-requisites satisfied, the next step is 'flow setup.' This phase involves preparing a list of test cases, setting up the options file and other tool specific settings. In this section some key setup details are explained.

2.2.1. Test cases preparation

In the context of this methodology, "test case" and "circuit under test" have a very specific meaning. "Circuit under test" is the analog block that will be tested against metrics of interest. "Test case" refers to a specific combination of inputs for which a specific metric is being tested. For example, in the digitally controlled amplifier example, the amplifier is the circuit under test and "test case" refers to one of the four different modes of operation.

As shown in the "flow setup" section of Figure 1, the list of checks to be performed on the target block is maintained in a file called "circuit parameter file," CPF. This file contains a list of alpha-numeric identifiers for every metric of interest. For the voltage amplifier example, the four gain settings are listed as four separate identifiers in the CPF file. This is a simple but important setup file that causes the flow to iterate over every test case until all the entries are processed.

Another file used in the flow is the "parameter file" discussed in section 2.1.3. This file, which is specific to a test case, contains a list of parameter/value pair in the SPICE format. The user needs to configure these parameter files values for every test case so that when it gets included from the SPICE netlist, the circuit (or the behavioral model in the case of co-simulation run) is configured to operate in the expected mode.

2.2.2. Post-processing and Co-sim Options Files

BMV flow requires two supporting files which set up co-simulation environment and post processing options for the automated checkers.

The co-simulation environment requires following options in the format of <option_name>=<option_value>:

```
- SUBCKTS = <analog sub-circuit block to be replaced>
- SPICE_TOP_LIB_CELL_VIEW = <behavioral model's view>
- SIGNAL_LIST = <the signal.lst  file>
```

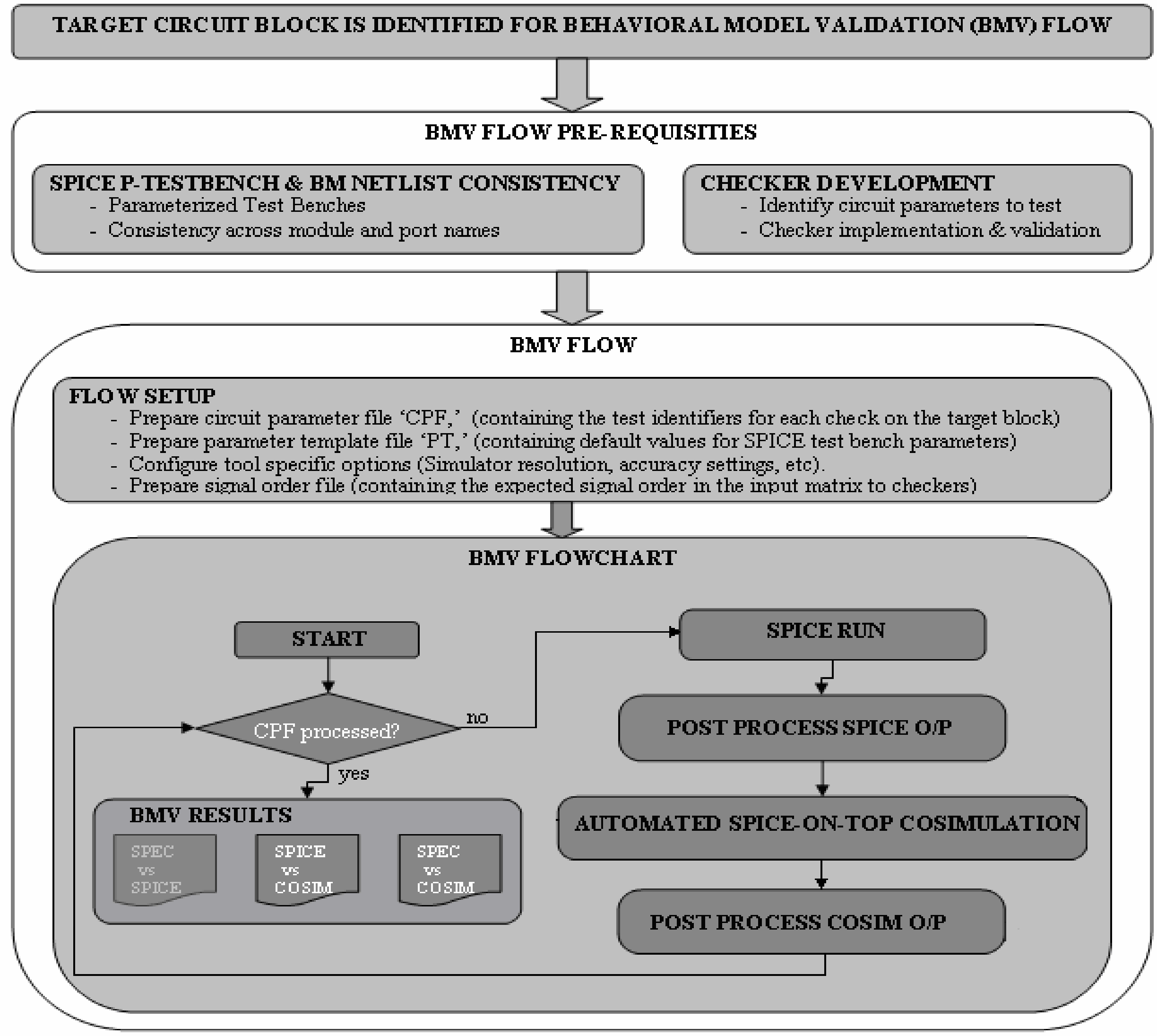

TARGET CIRCUIT BLOCK IS IDENTIFIED FOR BEHAVIORAL MODEL VALIDATION (BMV) FLOW
BMV FLOW PRE-REQUISITIES
SPICE P-TESTBENCH & BM NETLIST CONSISTENCY
- Parameterized Test Benches
- Consistency across module and port names
CHECKER DEVELOPMENT
- Identify circuit parameters to test
- Checker implementation & validation
BMV FLOW
FLOW SETUP
- Prepare circuit parameter file 'CPF,' (containing the test identifiers for each check on the target block)
- Prepare parameter template file 'PT,' (containing default values for SPICE test bench parameters)
- Configure tool specific options (Simulator resolution, accuracy settings, etc).
- Prepare signal order file (containing the expected signal order in the input matrix to checkers)
BMV FLOWCHART
START
CPF processed?
no
yes
SPICE RUN
POST PROCESS SPICE O/P
AUTOMATED SPICE-ON-TOP COSIMULATION
POST PROCESS COSIM O/P
BMV RESULTS
SPEC vs SPICE
SPICE vs COSIM
SPEC vs COSIM

Figure 1. Behavioral Model Validation (BMV) Flowchart.

The 'SUBCKTS' option specifies the analog block name to be replaced by its behavioral model. 'SPICE_TOP_LIB_CELL_VIEW' specifies the view (architecture name) to use from the compiled behavioral model library. 'SIGNAL_LIST' file specifies the full path to a file containing the order of the signals as expected by the checkers. This file specifies the signals of interest to the checker. These signals are extracted from the simulation output files and written out in the format expected by the checker. The 'ppOptions.opt' file is used by the post processing mechanism (more details in sub-section 3.2.3). The post processing options are:

```
- INPUT          = <test_ID>.fsdb.
- CHK_INPUT      = <test_ID>.tbl
- SIGNALS_LIST   = <the signal.lst file>
- REFERENCE      = ref_<test_ID>.txt
- CALCULATED     = calc_<test_ID>.txt
- THRESHOLD      = <%threshold for comparison>
- RESULTS        = results_<test_ID>.txt
- CHK_EXEC       = <checker command line>
```

The actual value of <test_ID> is the same as the entry in the CPF file. 'INPUT' specifies the simulation output file name. 'CHK_INPUT' is the ASCII file (in table format) name which serves as input to the checker. 'SIGNALS_LIST' must be the same as in 'cosim.options' file. 'REFERENCE' refers to the location of the reference file and 'CALCULATED' refers to the location of the file that contains the calculated values for the metric of interest. The format used in these files is: <test_ID> <numerical checker output>. 'THRESHOLD' option specifies the relative or absolute threshold to use when comparing the calculated value with the reference value. 'RESULTS' points to the location of results_<test_ID>.txt file that contains the result of the comparison. 'CHK_EXEC' option specifies the UNIX command line for invoking the automated checker. Directly using the command line for checker invocation allows the user to use any kind of checker independent of the implementation language used by the checker developer.

2.3. The BMV Flow

BMV Algorithm:

1. **All entries in circuit parameter file (CPF) processed?**
   1.1. **If yes, go to step 6.**
   1.2. **If no, go to next step 2.**

2. **Do SPICE run**

3. **Post process SPICE results**

4. **Do Co-simulation run (Invoke cosimPostProcess.pl with –COSIM switch)**

5. **Post process COSIM results**

6. **Print dashboards (results)**

7. **Stop.**

The flow performs two simulations for every test case. During SPICE simulation, results are compared with the SPEC (theoretical expectation). During SPICE-on-top co-simulation, results are compared with those from the SPICE simulation.

Figure 1 shows the steps involved in the BMV flow and Figure 2 describes the work flow for each test case (i.e. step 2 through 6 of the BMV algorithm) graphically. Three different comparisons are performed corresponding to the three ovals in Figure 2. These comparisons represent the three dashboards that are generated by the flow i.e., SPEC vs SPICE, SPICE vs COSIM, and SPEC vs COSIM.

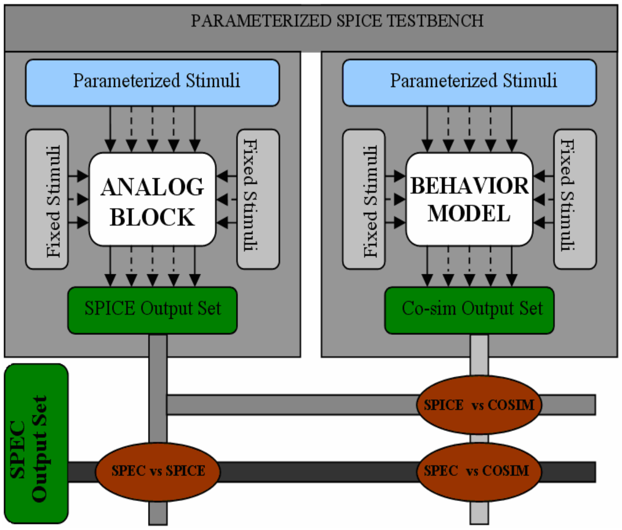


Figure 2. High level representation of BMV flow.

The parameterized testbench has two different sets of inputs and one set of outputs. The parameter file (PT) which contains the parameter/value pair for controlling inputs constitutes "parameterized stimuli" and the other inputs that are common across all test cases constitute "fixed stimuli".

Three sets of values for each metric are used in the flow i.e. SPICE, CO-SIM, and SPEC. SPICE and CO-SIM values are generated by the SPICE and co-simulation runs while SPEC values are theoretical expectations.

3. BMV FLOW CASE STUDY

3.1. The Digitally Controller Oscillator (DCO)

A DCO circuit is used as a test case for validating the BMV flow. The DCO is a low-voltage deep-submicron CMOS oscillator with 4, 574 transistors on C021 process technology capable of supporting 70 nm minimum gate lengths. It enables frequency synthesis without the use of any analog tuning voltage control line [9]. It operates in discrete time domain, although the underlying functionality is continuous time and continuous amplitude in nature [10]. The DCO is built using an LC oscillator with fixed inductance and variable capacitance.

The variable capacitance is implemented using MOS varactors. The varactor array is divided into three major groups that reflect three general operational modes: process-voltage temperature (PVT), acquisition (ACQ), and tracking (TFB).

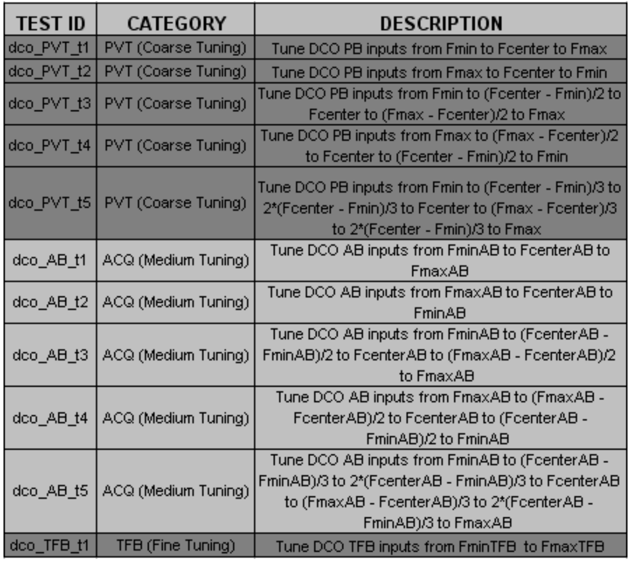
The first and second groups coarsely set the desired center frequency of oscillation before the actual transmit or reception begins, and the third group controls the oscillating frequency precisely during actual operation. PVT mode uses 7 bit binary weighted encoding. Each step of PVT is Δf^{PVT} = 2.0 Mhz, Acquisition is Δf^{ACQ} = 500 Khz and for tracking fractional (TFB) mode, it is $\Delta f^{TIB/TFB}$ = 35 KHz [10].

3.2. Metrics of Interests and BMV for DCO

The key care about for the DCO is the tuning precision. Based on the digital controls, the oscillating frequency of DCO should be predictable with certain degree of accuracy. Of paramount concern is the monotonicity of the frequency stepping in a given direction of traversal. The Analog Model Checklist for DCO is shown in Table I.

To verify this in simulation, a checker was implemented in PERL. The time value pair of the oscillator output is captured by the simulator and provided to this checker as a file pointer. Other inputs are the direction of traversal, period of transient simulation and the precise time stamp at which traversal is forced. As an example, the frequency steps for the PVT mode are shown in Figure 3.

Table I: DCO Analog Model Checklist



TEST ID	CATEGORY	DESCRIPTION
dco_PVT_t1	PVT (Coarse Tuning)	Tune DCO PB inputs from Fmin to Fcenter to Fmax
dco_PVT_t2	PVT (Coarse Tuning)	Tune DCO PB inputs from Fmax to Fcenter to Fmin
dco_PVT_t3	PVT (Coarse Tuning)	Tune DCO PB inputs from Fmin to (Fcenter - Fmin)/2 to Fcenter to (Fmax - Fcenter)/2 to Fmax
dco_PVT_t4	PVT (Coarse Tuning)	Tune DCO PB inputs from Fmax to (Fmax - Fcenter)/2 to Fcenter to (Fcenter - Fmin)/2 to Fmin
dco_PVT_t5	PVT (Coarse Tuning)	Tune DCO PB inputs from Fmin to (Fcenter - Fmin)/3 to 2*(Fcenter - Fmin)/3 to Fcenter to (Fmax - Fcenter)/3 to 2*(Fcenter - Fmin)/3 to Fmax
dco_AB_t1	ACQ (Medium Tuning)	Tune DCO AB inputs from FminAB to FcenterAB to FmaxAB
dco_AB_t2	ACQ (Medium Tuning)	Tune DCO AB inputs from FmaxAB to FcenterAB to FminAB
dco_AB_t3	ACQ (Medium Tuning)	Tune DCO AB inputs from FminAB to (FcenterAB - FminAB)/2 to FcenterAB to (FmaxAB - FcenterAB)/2 to FmaxAB
dco_AB_t4	ACQ (Medium Tuning)	Tune DCO AB inputs from FmaxAB to (FmaxAB - FcenterAB)/2 to FcenterAB to (FcenterAB - FminAB)/2 to FminAB
dco_AB_t5	ACQ (Medium Tuning)	Tune DCO AB inputs from FminAB to (FcenterAB - FminAB)/3 to 2*(FcenterAB - FminAB)/3 to FcenterAB to (FmaxAB - FcenterAB)/3 to 2*(FcenterAB - FminAB)/3 to FmaxAB
dco_TFB_t1	TFB (Fine Tuning)	Tune DCO TFB inputs from FminTFB to FmaxTFB

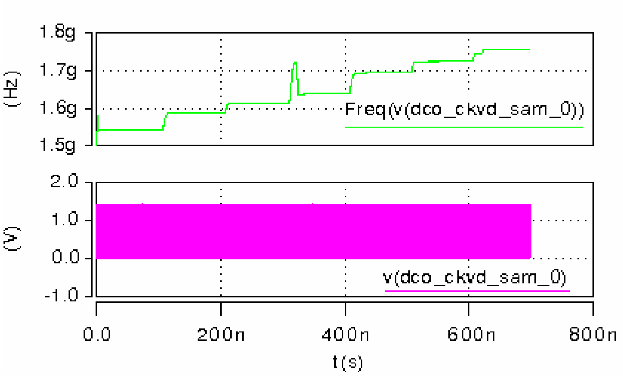


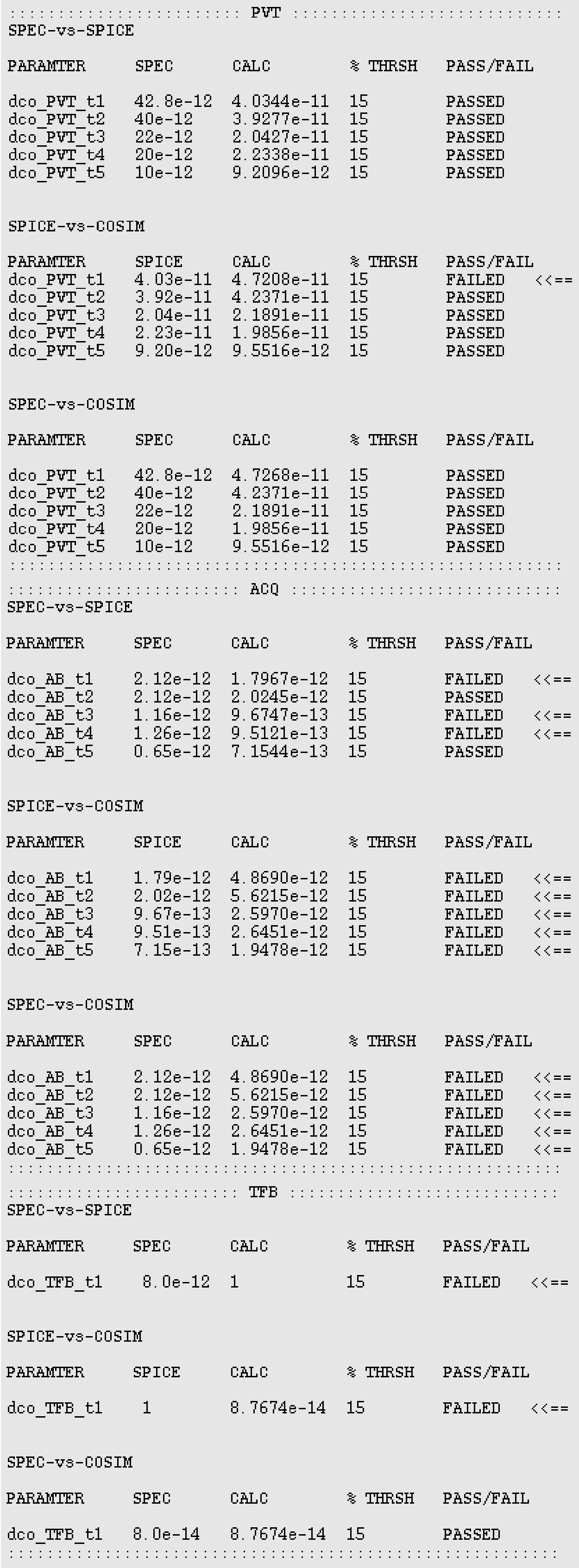
Figure 3. DCO PVT Frequency Stepping (Monotonic behavior)

The checker uses these inputs to compute the period of oscillation based on zero-crossings and the direction of traversal and provides this information in a text output. The BMV flow compares this with the golden reference values to flag an error in case of mismatch. The results of the DCO block are shown in Figure 4.

3.3. Analysis of DCO BMV Results

The PVT failures were traced to incorrect generic settings in the DCO behavioral model which limited the minimum frequency. ACQ failures pointed to incorrect bit mapping of the ACQ bits in the oscillator core. The flagged failures in the ACQ results helped amending the DCO spec.

The varactor sizes were modified to create a distinctive frequency response in the highest precision TFB mode. Frequency stepping range was 1600fs with new sizes. While the expectation is a time period deviation of 1600fs in the positive direction, the BMV flow determined that the results did not meet this expectation.



```
::::::::::::::::::::::: PVT ::::::::::::::::::::::::::
SPEC-vs-SPICE

PARAMTER     SPEC      CALC        % THRSH   PASS/FAIL

dco_PVT_t1   42.8e-12  4.0344e-11  15        PASSED
dco_PVT_t2   40e-12    3.9277e-11  15        PASSED
dco_PVT_t3   22e-12    2.0427e-11  15        PASSED
dco_PVT_t4   20e-12    2.2338e-11  15        PASSED
dco_PVT_t5   10e-12    9.2096e-12  15        PASSED


SPICE-vs-COSIM

PARAMTER     SPICE     CALC        % THRSH   PASS/FAIL
dco_PVT_t1   4.03e-11  4.7208e-11  15        FAILED   <<==
dco_PVT_t2   3.92e-11  4.2371e-11  15        PASSED
dco_PVT_t3   2.04e-11  2.1891e-11  15        PASSED
dco_PVT_t4   2.23e-11  1.9856e-11  15        PASSED
dco_PVT_t5   9.20e-12  9.5516e-12  15        PASSED


SPEC-vs-COSIM

PARAMTER     SPEC      CALC        % THRSH   PASS/FAIL

dco_PVT_t1   42.8e-12  4.7268e-11  15        PASSED
dco_PVT_t2   40e-12    4.2371e-11  15        PASSED
dco_PVT_t3   22e-12    2.1891e-11  15        PASSED
dco_PVT_t4   20e-12    1.9856e-11  15        PASSED
dco_PVT_t5   10e-12    9.5516e-12  15        PASSED
::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::: ACQ ::::::::::::::::::::::::::
SPEC-vs-SPICE

PARAMTER     SPEC      CALC        % THRSH   PASS/FAIL

dco_AB_t1    2.12e-12  1.7967e-12  15        FAILED   <<==
dco_AB_t2    2.12e-12  2.0245e-12  15        PASSED
dco_AB_t3    1.16e-12  9.6747e-13  15        FAILED   <<==
dco_AB_t4    1.26e-12  9.5121e-13  15        FAILED   <<==
dco_AB_t5    0.65e-12  7.1544e-13  15        PASSED


SPICE-vs-COSIM

PARAMTER     SPICE     CALC        % THRSH   PASS/FAIL

dco_AB_t1    1.79e-12  4.8690e-12  15        FAILED   <<==
dco_AB_t2    2.02e-12  5.6215e-12  15        FAILED   <<==
dco_AB_t3    9.67e-13  2.5970e-12  15        FAILED   <<==
dco_AB_t4    9.51e-13  2.6451e-12  15        FAILED   <<==
dco_AB_t5    7.15e-13  1.9478e-12  15        FAILED   <<==


SPEC-vs-COSIM

PARAMTER     SPEC      CALC        % THRSH   PASS/FAIL

dco_AB_t1    2.12e-12  4.8690e-12  15        FAILED   <<==
dco_AB_t2    2.12e-12  5.6215e-12  15        FAILED   <<==
dco_AB_t3    1.16e-12  2.5970e-12  15        FAILED   <<==
dco_AB_t4    1.26e-12  2.6451e-12  15        FAILED   <<==
dco_AB_t5    0.65e-12  1.9478e-12  15        FAILED   <<==
::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::: TFB ::::::::::::::::::::::::::
SPEC-vs-SPICE

PARAMTER     SPEC      CALC        % THRSH   PASS/FAIL

dco_TFB_t1    8.0e-12  1           15        FAILED   <<==


SPICE-vs-COSIM

PARAMTER     SPICE     CALC        % THRSH   PASS/FAIL

dco_TFB_t1    1        8.7674e-14  15        FAILED   <<==


SPEC-vs-COSIM

PARAMTER     SPEC      CALC        % THRSH   PASS/FAIL

dco_TFB_t1   8.0e-14   8.7674e-14  15        PASSED
::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

Figure 4. Dashboards showing DCO BMV results.

This failure was flagged as severe with code “1” (Figure 4). Figure 5 shows this response. After detailed investigation, it was determined that there was inversion added to the TFB bits in the SPICE netlist.

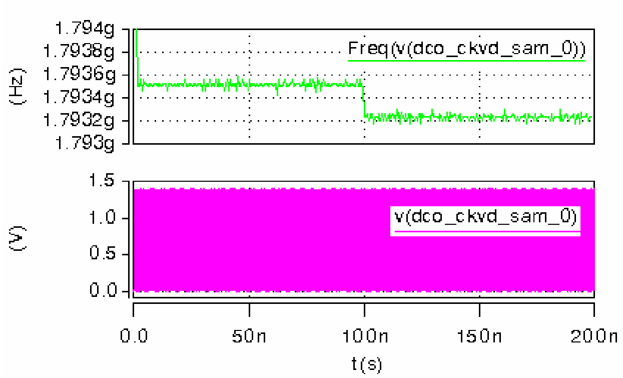


Figure 5. DCO TFB Frequency Stepping (non-monotonic).

3.4. Applicability to other Mixed-Signal circuits

Previous sections demonstrated the BMV methodology with a discrete circuit (Digitally Controlled Oscillator). The applicability of the BMV flow is not restricted to this category of circuits because the methodology is independent of the type of design being validated. This flexibility is offered by the use of custom checkers that are design specific and can be equipped to process both discrete and continuous simulation data to extract metrics of interest. The BMV flow was also used to validate the gain of an amplifier (continuous time output) against various gain combinations. The analysis of results pointed to a deficiency in the behavioral model due to which the output gain did not reflect changes in the bias currents at the input. It should be noted that deviations between SPICE and behavioral model can be caused by several other factors, such as insufficient or incorrect modeling or inability to model higher order effects of transistors. The goal of this methodology is to highlight these deviations. The user needs to correctly identify the root cause of the deviation and decide whether or not it makes sense to fix it by more accurate modeling.

4. ANALOG MODEL CHECKLIST

The key stakeholders in behavioral modeling are System Design, Circuit Design and Verification teams. System and Circuit Design teams are more focused on the performance of the model and how closely the model matches with the circuit. The Verification team is concerned about the modeling of controls, interfaces, supply voltages in many cases and power consumption modeling as well. The reference for each of these teams is the Design Specification. Each stakeholder defines key care abouts for every block to be modeled. These are captured in “Analog Model Template”. “Analog Model Checklist is created with owners and approvers. The Analog Model Template documents the detailed description of the modeling strategy and the effects being modeled along with the post processing used to evaluate the model and circuit performance. It also lists various stages of development and targets for each stage. Analog Model Checklist is used to track the completion of every stage and to ensure that model development and verification progress is being made in the right direction.

5. CONCLUSION

Real circuit bugs can easily escape even in successful SoC level simulations if the behavioral model does not correlate with the circuit and the circuit carries these bugs (for ex. the inversion bug in TFB mode of DCO). Clearly, the impact of such non-equivalence between model and circuit resulted in unexpected failure on silicon and thereby proved to be costly to the project in terms of time and human resource. BMV flow can provide a model to the SoC team which will accurately reflect the functional behavior of the SPICE circuit and in the process debug functional issues and address them early on in the project cycle. It enables the functional teams to ensure that all specification details are addressed in the model.

REFERENCES

[1]. K. Muhammad, T. Murphy and R. B. Staszewski “Verification of Digital RF Processors: RF, Analog, Baseband, and Software,” *IEEE Journal of Solid-state Circuits*, vol. 42, no. 5, pp. 992–1002, May. 2007.

[2]. W. Yang, “A high-level VHDL-AMS model design methodology for analog RF LNA and Mixer,” *IEEE Journal of Solid-state Circuits*, vol. 42, no. 5, pp. 992–1002, May. 2007.

[3]. R. Khouri et al, “Wireless System Validation using VHDL-AMS Behavioral Antenna Models: Radio-Frequency Identification case study,” *European Conference on Wireless Technology*, Amsterdam, 2004.

[4]. S. Joeres and S. Heinen, “Functional verification of RF SoCs using mixed-mode and mixed-domain simulations,” *Proceedings of the 2006 IEEE International Workshop on Behavioral modeling and Simulation,* pp.144-149, Sept. 2006.

[5]. B. Foret et al, “Unified Environment for Mixed-signal Top-level SoC Verification,” *IEEE Journal of Solid-state Circuits*, vol. 42, no. 5, pp. 992–1002, May. 2007.

[6]. A. Mounir et al, “Automatic behavioral model calibration for efficient PLL system verification,” *Proceedings of Design, Automation and Test in Europe*, DATE’03.

[7]. C. Visweswariah et al, “Model development and verification for high level analog blocks,” *25th ACM/IEEE Design Automation Conference,* paper 25.3, pp 376-382, 1988.

[8]. M. Sida et al, “Bluetooth transceiver design and simulation with VHDL-AMS,” *IEEE Circuits and Devices magazine,* pp 11-14, March 2003.

[9]. R.B. Staszewski, D. Leipold, K. Muhammad and P. T. Balsara “Digitally Controlled Oscillator (DCO) – Based Architecture of RF frequency Synthesis in a Deep-Submicrometer CMOS process” *IEEE transactions on Circuits & Systems– II: Analog & Digital Signal Processing, vol.50, NO.11, and Nov. 2003.*

[10]. R.B. Staszewski, D. Leipold, and P. T. Balsar “ALL–Digital Frequency Synthesizer in Deep-Submicron CMOS” *textbook, wILEY inter-science ISBN 0-471-77255-0.*