The Notion of Cross Coverage in AMS Design Verification

Sayandeep Sanyal[1], Aritra Hazra[1], Pallab Dasgupta[1],
Scott Morrison[2], Sudhakar Surendran[3], and Lakshmanan Balasubramanian [3]

[1]Dept. of Computer Science & Engineering, Indian Institute of Technology Kharagpur
[2]Texas Instruments, USA
[3]Texas Instruments (India) Pvt. Ltd.
{sayandeep.sanyal@, aritrah@cse., pallab@cse.}iitkgp.ac.in
{scott, sudhakars}@ti.com
lakshmanan@ieee.org

Abstract— Coverage monitoring is fundamental to design verification. Coverage artifacts are well developed for digital integrated circuits and these aim to cover the discrete state space and logical behaviors of the design. Analog designers are similarly concerned with the operating regions of the design and its response to an infinite and dense input space. Analog variables can influence each other in far more complex ways as compared to digital variables, consequently, the notion of *cross coverage*, as introduced in the analog context for the first time in this paper, is of high importance in analog design verification. This paper presents the formal syntax and semantics of analog cross coverage artifacts, the methods for evaluating them using our tool kit, and most importantly, the insights that can be gained from such cross coverage analysis.

I. Introduction

Existing coverage metrics for design validation are broadly divided into metrics for *code coverage* [1] and *functional coverage* [2], [3]. Metrics for code coverage [1], [4]–[6] includes coverage of statements, blocks, branches, toggles of nets, and FSM states, and the aim is to assess the part of the HDL code and/or the part of the state space that has been covered by simulation. Functional coverage metrics monitor the coverage of the functionality of the design, and therefore the coverage target is defined by the user in terms of syntactic features capturing the functional bins that must be explored during simulation. In SystemVerilog [7], functional coverage points are defined using the `coverpoint` definition, and multiple coverpoints can be grouped under a `covergroup`.

For analog designs, code coverage is not particularly useful since all parts of a typical analog design are functional at all times. The notion of functional coverage is also fundamentally different in analog verification, since the definition of functional correctness in analog is not based on logical artifacts, but on ensuring that functional relationships between analog variables (including inputs and outputs) stay within acceptable limits of tolerance. For example, the correctness of the output of a digital comparator can be defined logically as a Boolean function over its digital input bits, where as, the specification of an analog comparator is defined in terms of the real valued resolution defining the limits of its ability to correctly compare two input voltages that are very close to each other. For the digital comparator, a logical error is equally likely to affect inputs that are well separated and inputs that are close to each other. On the other hand, for the analog comparator, comparing input voltages that are close to each other is more significant and therefore more valuable from a coverage perspective. The definition of coverage bins for analog verification needs synergy with the verification concerns in the analog domain.

Often it is important to study the combinations of values taken by two or more variables at the same time. The notion of such *cross coverage* exists in terms of the `cross` construct defined inside SystemVerilog coverage specifications, but the role of *cross coverage* in AMS design verification has not been reported in the literature. As part of our effort towards building a CAD framework for coverage management in mixed-signal integrated circuit designs, we have worked on the basic coverage types for analog coverage and have studied the notion of cross coverage in the analog context. These studies led us to believe that cross coverage is not only germane to analog design verification, but has an impact on it which far surpasses the impact of cross coverage in the digital domain. This is because a large majority of analog features relate one analog quantity with another, and it is a combination of two or more variables that expose a design bug. Examples include voltage droops involving output voltage and load current, rise time / settling time violations involving input voltage and output voltage, peak overshoot involving output voltage and settling voltage, etc.

This paper focuses on cross coverage for AMS design verification. Section II outlines existing SystemVerilog support for cross coverage. Section III outlines some of our basic coverage types for AMS coverage management, specifically chosen to highlight the notion of cross coverage. The syntax for defining cross coverage is presented in Section IV. A collection of case studies on cross coverage is presented in Section V highlighting the relevance of cross coverage to an analog designer. Section VI presents experimental results.

This work was supported by Semiconductor Research Corporation under GRC Task ID 2810.019

II. Cross Coverage in SystemVerilog

To illustrate the existing notion of cross coverage in SystemVerilog, we consider the coverage of instructions executed in different operating modes of a CPU. Most CPUs have a well defined *kernel mode* and multiple *user modes* of operation. The CPU modes place restrictions on the type and scope of operations that can be performed by certain processes being run by the CPU. For simplicity, we consider a CPU with 8-bit opcodes (256 instructions) and 2-bit mode (4 modes). Consider the following `covergroup` definition in SystemVerilog:

```
input bit clk, decode,
input logic [7:0] opcode,
input logic [1:0] mode;
covergroup cg
   @(posedge clk);
      coverpoint opcode;
      coverpoint mode;
      cross opcode, mode;
endgroup
cg cg_Inst = new;
```

The coverpoint `opcode` defines 256 coverage bins for checking which instructions are executed. The coverpoint `mode` defines 4 coverage bins for checking which operating modes are visited during simulation. This coverage information does not show which instructions are executed under each mode of execution, all though that is an area of concern from a verification perspective. Cross coverage enables us to find this information.

The `cross` between opcode and mode creates 256×4 cross coverage bins for monitoring the $\langle opcode, mode \rangle$ combinations encountered during simulation. If some instruction is taboo in some operational mode, then the corresponding cross coverage bin should remain empty.

III. AMS Coverage Primitives

Cover points in the digital context have enumerated domains, but the dense domains of analog variables require coverage bins to be defined as intervals.

Definition 1. [Bin:]
A bin, β, can be defined as a non-empty convex subset of $\mathbb{R}$, expressed as $[a : b]$, $(a : c)$, $[a, c)$ and $(a, c]$ where $a, b, c \in \mathbb{R}$ and $b \geq a, c > a$. Here, a, b and c are called bin boundaries*; $l(\beta)$ and $r(\beta)$ are used to denote the left and right boundaries of bin β, respectively.* □

The input function is defined as a time-stamped data of the signals for which the coverage is to be computed.

Definition 2. [Input Function:]
The Input Function, μ, is defined as a mapping, $\mu(v, t) : \mathbb{R}^+ \rightarrow \mathbb{R}$ for a signal v and time t. Similarly, for a system having the set of signals as, $\overrightarrow{v} = \{v_1, v_2, \ldots, v_k\}$, the mapping is defined as, $\mu(\overrightarrow{v}, t) : \mathbb{R}^+ \rightarrow \mathbb{R}^k$. □

Definition 3. [Simulation Trace]
A simulation trace, τ, is a mapping $\tau : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^{|V|}$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of variables (Boolean and Real) representing signals of the system and $|V|$ indicates the cardinality of the variable set. □

We outline some of the AMS coverage primitives introduced and supported by our tool [1]. We shall use them later in the context of cross coverage. These represent only a subset of the primitives supported by our tool.

Coverage Primitive 1. [Range Coverage]
Given a signal v, the range coverage returns an interval $[a, b]$ ($a, b \in \mathbb{R}$ and $a \leq b$) such that $\forall x,\ x \in \mathtt{range}(\mu(v, t))$, when $a \leq x$ and $b \geq x$. Since all the signals are continuous, a range coverage of $[a, b]$ in a single simulation run indicates that the signal v has covered the set of bins*:*

$$\mathcal{B}^{v}_{rangeCov} = \{\ [a : a + k], (a + k : a + 2k], \cdots, [b - k : b]\ \}$$

where $k \in \mathbb{R}^+$ is the value of a user defined parameter binGranularity*. The syntax for defining range coverage is as follows:*

```
covergroup <coverGroupName>
   -range{
      -binGranularity <real_value>
      -valTol <real_value>
      -timeTol <real_value>
   }
endgroup
```

The value tolerance and time tolerance parameters define the resolution with which the bin boundaries are monitored by our tool.

Coverage Primitive 2. [De-glitched Range Coverage]
Designers are often interested in knowing the range of a signal after deglitching it from transients and noise. For identifying a glitch in a signal, v, a user-defined parameter deglitchingTime *is proposed. A behavior of v, in the time interval $[t_1 : t_2]$, is identified as a* glitch *if and only if the following conditions apply:*

1) *If $\exists x$ such that x is a bin boundary, v has positive and negative crossings of x at time t_1 and t_2 respectively, or vice-versa.*
2) *$t_2 - t_1 \leq \delta_t$.*

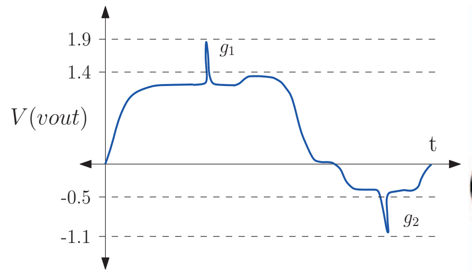
where δ_t is given as the parameter deglitchingTime. The syntax for defining deglitched range coverage is as follows:

```
covergroup <coverGroupName>
   -deglitch{
      -binGranularity <real_value>
      -deglitchingTime <real_value>
      -valTol <real_value>
      -timeTol <real_value>
   }
endgroup
```

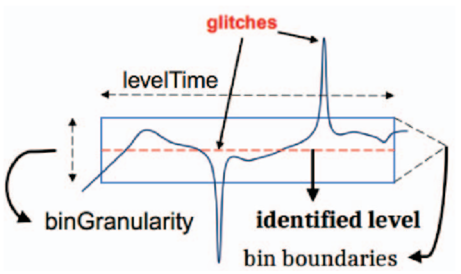
In Figure 1a, the spike (g_1) and the trough (g_2) are identified as glitches. Considering *binGranularity* of 0.1, the range coverage and de-glitched range coverage are reported as $[-1.1 : 1.9]$ and $[-0.5 : 1.4]$ respectively.

Coverage Primitive 3. [Level Coverage]
Level Coverage reports the discrete levels where a signal has settled for at a specified amount of time. The attributes of

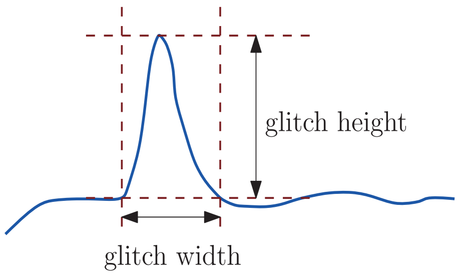
[1]Refer to Section VI and Figure 4



(a) Range Coverage



(b) Level Coverage



(c) Glitch Coverage

Fig. 1: Some AMS Coverage Primitives

level coverage are binGranularity *k*, deglitchingTime *δ_t, and* levelTime*, δ_l. The signal is deglitched with the values of k and δ_t. Following that, l_i is called a* level *of signal v if and only if $\exists \hat{t}$ such that $\forall t' \in [\hat{t} : \hat{t} + \delta_l]$, $\mu(v, t') \in [l_i - \frac{k}{2} : l_i + \frac{k}{2}]$. Figure 1b shows the necessary conditions for a signal to be identified at a discrete level. Thus the level coverage artefact returns a set of discrete values of different* levels*. The syntax for defining level coverage is as follows:*

```
covergroup <coverGroupName>
   -level{
      -binGranularity <real_value>
      -deglitchingTime <real_value>
      -levelTime <real_value>
      -valTol <real_value>
      -timeTol <real_value>
   }
endgroup
```

Coverage Primitive 4. [Glitch Coverage] *A glitch coverage monitor captures the glitches in a waveform. If the portion of a signal v within the interval $[t_1 : t_2]$ has been identified as a glitch g_i, then the dimension of the glitch is defined by its height, $x_{g_i} = \mathtt{max}\{\ \mu(v, t') : \forall t' \in [t_1 : t_2]\ \} - \mathtt{min}\{\ \mu(v, t') : \forall t' \in [t_1 : t_2]\ \}$, and its width, $t_{g_i} = t_2 - t_1$, where $x_{g_i} \in \mathbb{R}$ and $t_{g_i} \in \mathbb{R}^+$ as shown in Figure 1c. The formal syntax for defining glitch coverage is as follows.*

```
covergroup <coverGroupName>
   -glitch{
      -binGranularity <real_value>
      -deglitchingTime <real_value>
      -timeGranularity <real_value>
      -valTol <real_value>
      -timeTol <real_value>
   }
endgroup
```

For declaring coverage on a certain signal, we construct *coverObject* from these covergroups with the name of signal as a parameter.

```
<coverObjectName> =
   new <coverageGroupName>{<signalName>};
```

IV. Cross Coverage in AMS

A cross coverage is a combination of multiple functional coverage artifacts. It can be a combination of the same coverage artifact over different signals, such as range of line voltage versus range of output voltage in a Low Dropout Linear voltage regulator (LDO). We may also be interested in different coverage artifacts over the same signal (such as load current versus voltage on the output line of a LDO). Sometimes we may have both combinations.

For a combination of n such functional coverage artefacts viz., $C_1, C_2, C_3, \ldots, C_n$, the result is a set of n-dimensional bins, $\mathcal{B}_{crossCover} \subseteq \mathcal{B}_{C_1} \times \mathcal{B}_{C_2} \times \mathcal{B}_{C_3} \times \cdots \times \mathcal{B}_{C_n}$, where $\mathcal{B}_{C_i}$ is the set of coverage bins of functional coverage artifact C_i.

Let $\mathcal{B}^\tau : \mathbb{R}^+ \times C_i \rightarrow \mathcal{B}_{C_i}$ denote that in a simulation trace τ, the functional coverage artifact C_i evaluates to a value (for a given time point) from the bin present in $\mathcal{B}_{C_i}$. Then:

$$\mathcal{B}_{crossCoverage} = \{\ \{\beta_1^j, \beta_2^j, \beta_3^j, \ldots, \beta_i^j, \ldots, \beta_n^j\} \mid \\ \forall j\ \exists \hat{t} \in dom(t),\ \forall i \in [1, n],\ \mathcal{B}^\tau(\hat{t}, C_i) = \beta_i^j \in \mathcal{B}_{C_i}\ \}$$

The syntax for specifying cross coverage within a covergroup is as follows:

```
<crossCoverageName> =
  cross {<coverageObject>,
                <coverageObject>, ...}
```

A. Cross coverage with Time

Just as we may have coverage bins for various intervals of voltage and current of a net, we may also have bins for time. Time is global and uniform for all signals, hence cross coverage with time bins require special attention.

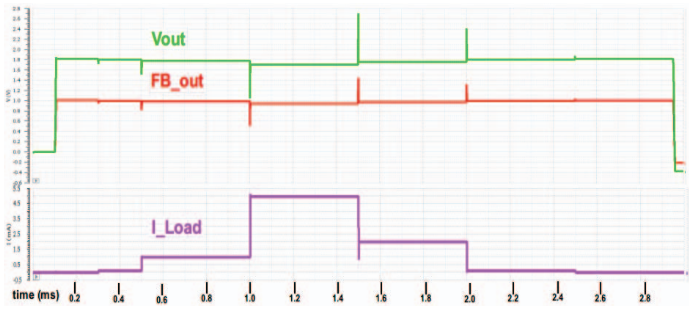
Definition 4. [Truth Intervals of Coverage Artifact] *The truth intervals of a coverage artifact, C, in a bin, β, for a simulation trace, τ, is defined as the set of time intervals, $\mathcal{I}_C^\tau(\beta) = \{\ [p_i : q_i] \mid p_i, q_i \in \mathbb{R}^+, p_i \leq q_i, \forall t \in [p_i, q_i]\ \mathcal{B}^\tau(t, C) = \beta\ \}$.* □

For a given functional coverage artifact, C, and a bin granularity (a real value) for time as $k_t \in \mathbb{R}^+$, its cross coverage with time can be classified into two sub-categories as follows.

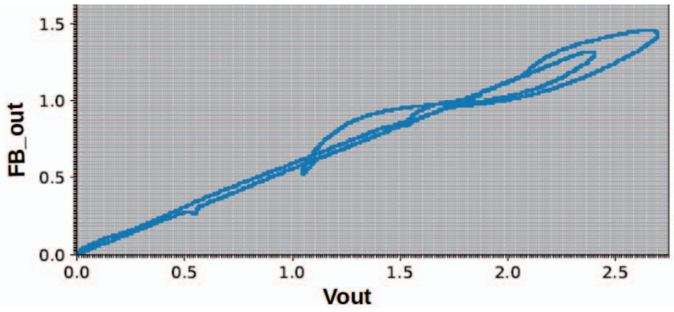
- **Total Time spent in a Bin**: The total time spent by C in a bin, β, for a simulation trace, τ, is given as, $\mathcal{I}_{total} = \sum_{i=1}^{|\mathcal{I}_C^\tau(B)|} (q_i - p_i), [p_i : q_i] \in \mathcal{I}_C^\tau(B)$.
- **Length of Disjoint Time Intervals spent in a Bin**: The length of each time interval in $\mathcal{I}_C^\tau(B)$ is also a coverage item and returns a set of discrete values: $\{\ (q_i - p_i) \mid [p_i : q_i] \in \mathcal{I}_C^\tau(B)\ \forall i \in [1, |\mathcal{I}_C^\tau(B)|]\ \}$.

B. Cross coverage with Operating Mode

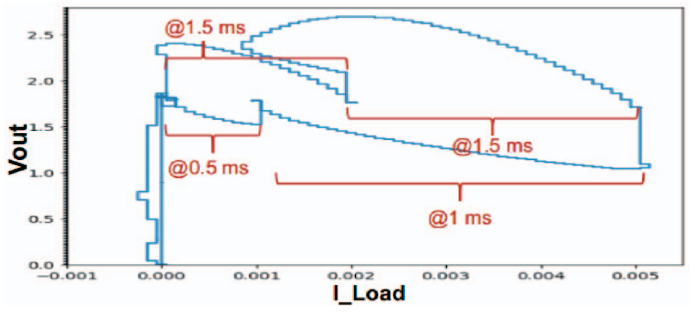
Coverage analysis in AMS designs is intricately related with the operational modes of a circuit. Often the mode of an analog



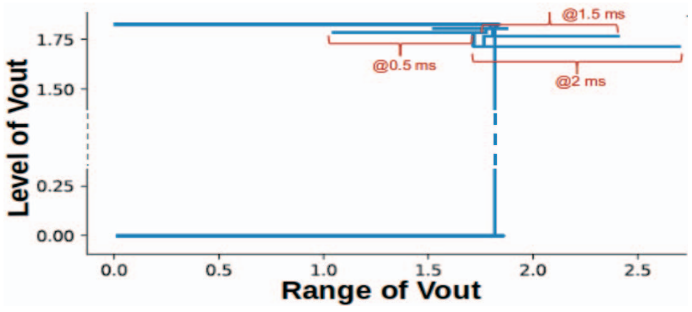
(a) Input Waveforms



(b) Cross coverage between Output Voltage and Feedback Voltage



(c) Cross coverage between Output Voltage and Load Current



(d) Cross coverage between Range and Level of output voltage

Fig. 2: Diagrams of Waveform and Different Case Studies on Cross Coverage

circuit is not explicitly provided by the circuit, but can be inferred by monitoring the guard conditions that stand between the modes. This can be done by defining an *auxiliary state machine*, which is co-simulated with the circuit. For example, a snippet of the auxiliary state machine for a LDO is shown below.

```
module LDO_ASM (state);
output state;
electrical state;
parameter integer SHUTDOWN = 0;
parameter integer STARTUP = 1;
parameter integer REGULATORY = 2;
parameter integer SHORTCKT = 3;
parameter real isc = 0.0085;
real STATE;
analog begin
   @(initial_step) STATE = SHUTDOWN;
   case(STATE)
      SHUTDOWN:
         if (V(xLDO.vin)>= 2.2)
                    STATE=STARTUP;
      STARTUP:
         if (V(xLDO.vin)< 2.2)
                   STATE=SHUTDOWN;
         else if(V(xLDO.vout)> 1.65)
                     STATE=REGULATORY;
      REGULATORY:
         if (V(xLDO.vin)< 2.2)
                  STATE=SHUTDOWN;
         else if (I(xLDO.vout)> isc)
                       STATE=SHORTCKT;
      SHORTCKT:
         if(V(xLDO.vout)< 0.2)
              STATE=SHUTDOWN;
   endcase
   V(state) <+ transition(STATE,0,0);
end
endmodule
```

Once the auxiliary state machine is defined, we use the enumerated variable `state` and define coverpoints for each value of STATE. Thereafter we can define cross coverage between STATE and other artifacts.

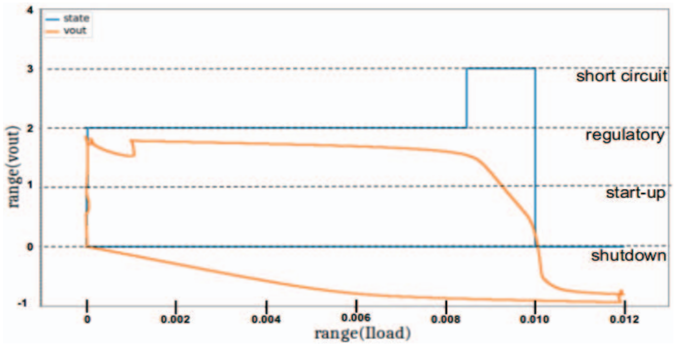
V. Illustrative Case Studies

This section presents case studies on various types of AMS cross coverage artifacts over the simulation output of an industrial LDO. The waveforms in Figure 2a shows the output voltage, *Vout*, feedback voltage, *FB_out*, and the load current, *I_Load*, over a period of time. During this period, the LDO starts up, and enters the regulatory mode with $Vout = 1.8V$. The load current increases (almost) discretely at the time points 0.5 sec and 1.0 sec, and drops at the time points 1.5 sec and 2.0 sec. The task of the LDO is to maintain the output voltage at the rated voltage under varying load currents.

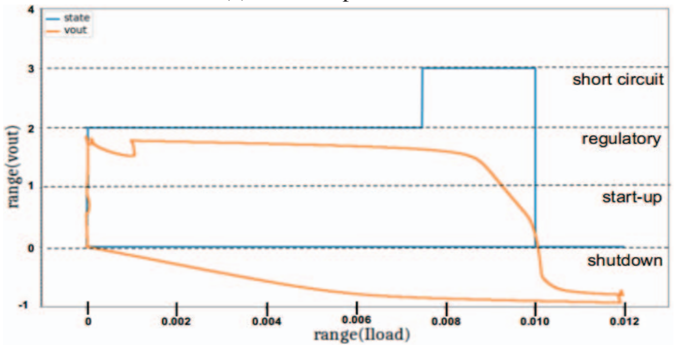
A. Cross between Output Voltage and Feedback Voltage

Figure 2b shows the cross coverage between the output voltage, *Vout*, of the LDO and the feedback voltage. The feedback voltage is derived from the output voltage in linear proportionality, and this voltage is used by the feedback control loop of the LDO to maintain its rated output voltage. The grid lines in Figure 2b shows the discretization of the continuous space using the user specified granularity (in this case, $0.01V$ for both the axes). Each cell in the grid represents a single coverage bin in the two dimensional space. The coloured cells represent the bins that have been covered during this simulation. The cross coverage analysis brings out the following insights, which are not very apparent from the input waveforms:

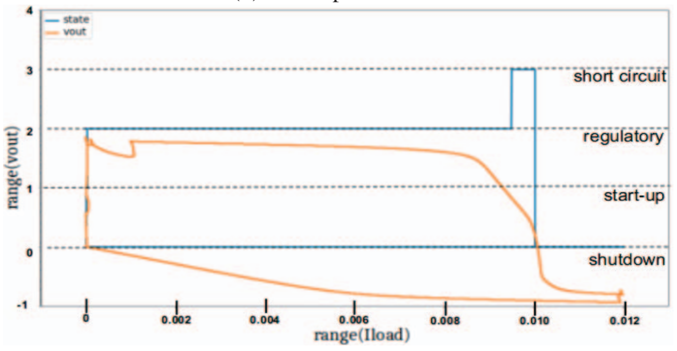
(A) As *Vout* rises, *FB_out* also follows it by maintaining a constant slope, namely the linear proportionality mentioned above.



(a) Actual Specification



(b) Safe Specification



(c) Unsafe Specification

Fig. 3: Application of cross coverage in validating Short Circuit current specification

(B) Both of the signals settle down at the stable operating region where *Vout* and FB_out have values of about 1.8 and 0.9 respectively. Hence this region is densely populated with the bins, as the LDO tries to maintain this operating point.

(C) Whenever *Vout* increases or decreases from this stable operating region (due to droops or surges), *FB_out* also increases or decreases respectively, which in turn ensures that the *Vout* signal returns to its operating region.

(D) It can also be seen that during this simulation *Vout* rose to a maximum voltage of around 2.8V which caused *FB_out* to rise as high as 1.4V. This represents the peak overshoot.

B. Cross between Output Voltage and Load Current

Figure 2c shows the cross coverage between *Vout* and *I_Load*. This illustrates the standard load regulation in an LDO, i.e., how the output voltage behaves with variations in load current. The lower parts of the loop shows the voltage droops and how the output voltage climbs up to rated voltage while supplying the increased load current. The upper portions of the loop shows the output surges as the load current drops sharply. It can also be seen that the LDO settles down at a sightly lower value when it is supplying a higher value of load current.

C. Cross between Level and Range of Output Voltage

Figure 2d shows the cross coverage between the range and level of signal the same signal, namely *Vout*. Since by definition, a voltage *level* is reached only when the circuit spends a specified interval of time in that voltage (with some amount of tolerance), this cross coverage essentially shows the size of the glitches encountered in each *level* that is covered. It may also be noted that though the rated output voltage is 1.8V, the LDO settles at several levels close to this value, depending on the load.

D. Coverage of Short Circuit Behavior

Typically, the specification of a LDO provides a conservative value for the short circuit current, and the LDO actually goes into short circuit mode and shuts down at a higher load current. When we replace the LDO with a behavioral model (BM) for full chip simulation, we must ensure that the short circuit current in the BM matches that of the netlist. For this, we use the auxiliary state machine (ASM) shown in Section IV-B. Figure 3a, Figure 3b, and Figure 3c show the cross coverage between the range of *Vout*, the range of *I_Load*, and the mode of the ASM. Figure 3a represents the case where the threshold used in the ASM for moving into the short circuit mode is accurate. Figure 3b represents the case when the threshold used in the ASM is conservative, and Figure 3c represents the case where the ASM overestimates the short circuit current. A BM should not use the threshold of Figure 3c.

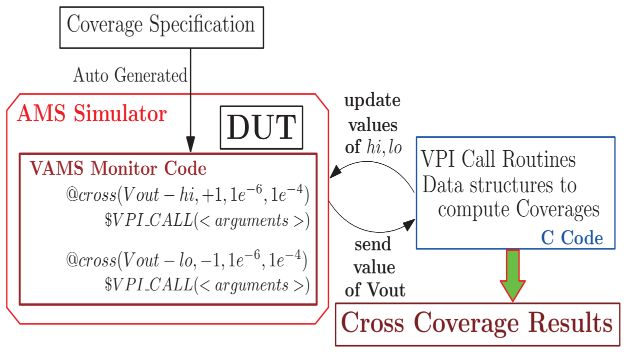

Coverage Specification
Auto Generated
AMS Simulator
DUT
VAMS Monitor Code
$@cross(Vout - hi, +1, 1e^{-6}, 1e^{-4})$
$\$VPI\_CALL(< arguments >)$
$@cross(Vout - lo, -1, 1e^{-6}, 1e^{-4})$
$\$VPI\_CALL(< arguments >)$
update values of hi, lo
send value of Vout
VPI Call Routines
Data structures to compute Coverages
C Code
Cross Coverage Results

Fig. 4: Tool Flow for Cross Coverage Computation

VI. Implementation and Results

Our toolkit interfaces with commercial simulation platforms through VPI callbacks (see Figure 4). The coverage monitoring can be done online during simulation, or offline by replaying the waveforms collected during simulation. For standard range coverage, the bin boundary crossings are detected using cross event monitors of Verilog AMS. Instead of creating a separate monitor for each coverage bin, the cross event monitors are dynamically reconfigured to the boundaries of the bin containing the present state of the circuit.

Table I shows the overhead of coverage computation over simulation time. Column 2 indicates the number of circuit

Circuit	#Circuit Element	Simulation Time	Coverage Monitoring Time								
			BG = 0.1			BG = 0.01			BG = 0.001		
			#cross	online	offline	#cross	online	offline	#cross	online	offline
LDO1	98	0.6s	60	1.2s	0.1s	616	3.5s	0.4s	6200	18.8s	2.8s
LDO2	176	5.4s	60	8.5s	0.1s	616	32.8s	0.4s	6200	3m 9s	2.8s
Buck Regulator	3412	1h19m	4116	1h26m	2.4s	15142	1h38m	10s	131729	3h7m	1m11s
Comparator Dump	NA	NA	140	NA	0.6s	432	NA	0.8s	3596	NA	2s

TABLE I

elements in each design, including transistor, resistor, capacitor, diode, voltage and current sources. We present results for three different choices for bin granularity (BG), and for each, we compare the circuit simulation time with online and offline coverage monitoring. The first three test cases are industrial test cases, whereas for the comparator we only had the waveform dumps from the industrial test case, and hence only offline coverage results are reported. The following observations are notable.

1) The percentage overhead of coverage monitoring rapidly declines with the size of the circuit as the simulation time dominates the total time.
2) It is more profitable to do the coverage monitoring offline after simulation. Adding the overhead of offline coverage monitoring with the native simulation time yields a figure which is less than the time for online coverage monitoring. This is because online coverage monitoring with high precision in the cross event monitors enforces additional simulation points, causing overhead.
3) The overhead increases with higher precision in BG, since more simulation points are added by the cross event monitors. We are currently working on adaptive bin boundaries to address this issue.

VII. Related Work and Conclusion

Several methodologies have been presented in literatures for verification of AMS systems. One line of work has been in building these methodologies around assertions designed for AMS systems. In [14] authors put forward an approach for checking assertions in analog domain by abstracting out the real-valued signals to binary predicates by defining Predicated Over Real Value on them. An assertion monitoring tool has been introduced in [13] which analyzes output signals to compute truth of assertions written in a language which incorporates Timed Regular Expression into Signal Temporal Logic. Others have focussed on verifying AMS systems by computing feature values relevant to AMS domain. The computation is done alongside circuit simulation [11] or over a hybird automata model of the system [12]. In recent times researchers have started to build up verification techniques over the coverage of AMS system. A method for discretizing the analog state space and defining coverage as the percentage of discrete states visited during simulation is presented in [8]. Analog *code coverage* metrices have been proposed in [1] for behavioral models in Verilog-A/AMS. A methodology using affine arithmetic and symbolic ranges has been outlined in [9]. In [10], abstract behavioral models have been used for achieving a fast estimate of functional coverage. All these methodologies either require a very high level abstraction of the system, or do not produce results in the parlance of analog designers. The notion of cross coverage for analog artifacts has been addressed for the first time in this paper. Through this exposition, we have also outlined the support of our tool kit and the methodology used for monitoring cross coverage.

References

[1] A.Fürtig, et al. ”Comparing Code Coverage Metrics for Analog Behavioral Models”, *IEEE 14th SMACD 2017.*

[2] B. Sahu, et al. ”Configuration-based Merging of Coverage Data Results for Functional Verification of Integrated Circuits”, *Oct 2015 US Patent 8,560,985.*

[3] S. Park, et al. ”A Functional Coverage Metric for Estimating the Gate-level Fault Coverage of Functional Tests”, *IEEE ITC, 2006.*

[4] C. Stoucny, et al. ”Alpha 21164 Manufacturing Test Development and Coverage Analysis”, *IEEE Design and Test of Computers, 1998.*

[5] F. Fallah, et al. ”Simulation Vector Generation from HDL Descriptions for Observability-enhanced Statement Coverage”, *DAC 1999.*

[6] H Chockler, et al. ”Coverage Metrics for Formal Verification” in *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, Springer 2003.

[7] ”IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language”, *IEEE Std 1800-2017.*

[8] A.Fürtig, et al. ”Feature Based State Space Coverage of Analog Circuits” in *2016 Forum on Specification and Design Languages.*

[9] E. Barke, et al. ”Embedded Tutorial: Analog-/Mixed-Signal Verification Methods for AMS Coverage Analysis”, *DATE 2016.*

[10] M. Horowitz, et al. ”Fortifying Analog Models with Equivalence Checking and Coverage Analysis”, in *Proc. of DAC 2010.*

[11] A. Ain, et al. ”Feature Indented Assertions for Analog and Mixed-Signal Validation”, in *IEEE TCAD Nov. 2016.*

[12] A. A. B. Costa, et al. ”Formal Feature Interpretation of Hybrid Systems”, in *IEEE TCAD Nov. 2018.*

[13] D. Nikovi, et al. ”AMT 2.0: Qualitative and Quantitative Trace Analysis with Extended Signal Temporal Logic”, in *TACAS 2018.*

[14] R. Mukhopadhyay, et al. ”Instrumenting AMS Assertion Verification on Commercial Platforms”, in *ACM TODAES 2009*