Analog Coverage-driven Selection of Simulation Corners for AMS Integrated Circuits

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

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

***Abstract*—Integrated circuit designs are evaluated at various corners defined by choices of the design and process parameters. Considering the large number of corners and the simulation cost of covering all the corners of a large design, it is desirable to identify a subset of the corners that can potentially expose corner case bugs. In an integrated analog coverage management framework, this choice may be influenced by those corners that take one or more component analog IPs close to their individual specification boundaries. Since the admissible state space of an analog IP is multi-dimensional, the same corner may not reach the extreme behaviors for each attribute of the specification, and one needs to identify a subset that covers the extremality. This paper shows that the underlying problem is NP-hard and presents an automated methodology for selecting the corners. A formal analog coverage specification is leveraged by our algorithm, which uses a Satisfiability Modulo Theory (SMT) solver to identify the appropriate corners from the output of multiple Monte Carlo (MC) simulations. The efficacy of the proposed approach is demonstrated over industrial test cases.**

I. Introduction

The design and verification of Analog and Mixed-signal (AMS) IPs and their integration into System-on-Chip (SoC) designs follow a bottom-up approach. The basic IP components are designed, verified, and then hierarchically integrated into bigger components, all the way up to the SoC level [1]–[3]. Today, we seek the reuse of design IPs in multiple designs, facilitated by the seamless integration of IPs in new SoCs with incremental verification needs.

The feasibility of covering all IP-level simulation coverpoints decreases as one goes up the design hierarchy. When IP-level corners are not exercised post IP integration, corner-case bugs are missed in pre-silicon validation and show up during post-silicon validation, where it is significantly more expensive to fix. Moreover, a bug caught in later stages may invalidate a portion of verification results that have been collected so far, and consequently, many verification steps may have to be repeated before the final sign-off of the design. Researchers have demonstrated methodologies to find a set of design parameter values that may lead to a faulty operation of the design. For example, the approach presented in [4], [5] uses Bayesian optimization for searching for such points of failure.

Several researchers have targeted the challenges faced during IP integration phase. An XML-based language format, Analog Specification Description in XML (ASDeX), was proposed in [6] to describe the specification of analog circuits along with functional behaviours, testbenches, and simulation settings for enabling automatic validation of IPs. IEEE 1685 IP-XACT standard [7] has been proposed to facilitate such IP integration. It offers the scope of describing meta-data of the IPs and their interconnections in a structured fashion in an XML format. The newer versions of the standard have been extended to include AMS IPs, power management statistics in the form of Common Power Format (CPF) and Unified Power Format (UPF), descriptions of area for physical design planning, etc. [8] elaborates on how IP-XACT can be used during the integration phase of industrial designs, while [9] fixes issues of parameter referencing that may arise towards re-usability of IPs.

Designers and verification engineers rely on random (or constraint-random) simulation environments that carry out multiple simulations by randomly choosing input parameters from a given state space. One such widely practised technique is Monte Carlo simulation [10]. A drawback of this technique is that less probable corners are missed, and out of the randomly chosen input vectors, many may not cause extremal output responses of the circuit.

We believe that recent work on analog coverage management [11]–[13] can be utilised effectively in this context to judiciously choose the coverage targets post IP integration. This paper proposes an approach that utilises the coverage data available from simulations at a lower level of the design hierarchy for shaping the coverage plan at higher levels of the design hierarchy. The main contributions of this paper are as follows:

1) We address the problem of leveraging the tests driving a design IP to its extremal behaviours to choose the combinations of simulation and process parameters that are necessary at the higher levels of the design hierarchy. The Monte Carlo process parameter values are extracted from the simulations selected on the basis of covering extremal behaviours, and these values are used to run targeted simulations after IP integration to exercise the same cover points. The proposed approach provides the formal basis for choosing the right combination of parameter values for exercising the coverage goals at the higher levels based on the coverage information at the IP level.
2) We show that the underlying problem of choosing the minimum set of tests that cover all the extremities is NP-

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

978-3-9819263-7-8/DATE23/© 2023 EDAA

hard. We provide a Satisfiability Modulo Theory (SMT) formulation for finding the optimal solutions to the problem.

3) In some cases the number of corners needed to cover all the extremities can be prohibitive, and hence one needs to arrive at a tradeoff. We present a methodology where we put an upper bound on the number of simulations (say k) that can be run at the higher level of the design hierarchy. If no solution to the SMT exists with a subset of k runs, then the coverage targets are progressively relaxed until a solution is found.

The rest of the paper is organised as follows. Section II gives a formal description of the problem statement. Section III proves the NP-hardness of the problem. Section IV presents our methodology. Section V presents the results and Section VI concludes the paper.

II. Problem Formulation

We are given:

- $X = \{x_1, x_2, \cdots, x_p\}$: set of p Monte Carlo parameters
- $S = \{s_1, s_2, \cdots, s_n\}$: set of n Monte Carlo simulations where each s_j has end-time T_j.
- $\Gamma : (X \times S) \rightarrow \mathbb{R}$ is a mapping, where $\Gamma(x_i, s_j)$ denotes the value of Monte Carlo parameter, x_i, used in simulation, s_j.
- $X^j = \Big\{\Gamma(x_1, s_j), \Gamma(x_2, s_j), \cdots, \Gamma(x_p, s_j)\Big\}$: values of all Monte Carlo parameters in simulation s_j.
- $\Delta = \Big\{X^j \mid \forall j \in [1, n]\Big\}$
- $C = \{C_1, C_2, \cdots, C_m\}$: set of m coverpoints that are monitored across the n simulations
- g_i: *goal condition* for coverpoint C_i

A *coverpoint* captures a functional verification intent of the design. The notion of *coverpoints* has recently been introduced in the analog context [13], and it includes artefacts like *range, level, frequency, delay*, etc. Since circuit behaviour changes with different values of the Monte Carlo parameters, a *coverpoint* may reach different values in different simulations depending on the value of X, even if the stimulus is the same.

Definition 1. [Value of a Coverpoint w.r.t. a Simulation]*: The value function V for a coverpoint at an intermediate time point of a simulation can be defined as a mapping, $V : (C \times \Delta \times \mathbb{R}^+) \rightarrow \mathbb{R}$. For a specific simulation, s_j, at time t, the value of coverpoint C_i is given by $V(C_i, X^j, t)$. The set of values that C_i accumulates in s_j over time is denoted as $C_i^j = \{V(C_i, X^j, t) \mid \forall t \in [0 : T_j]\}$* ■

Definition 2. [Coverpoint Extremality]*: We define $min(C_i^j)$ and $max(C_i^j)$ as the extremal values for coverpoint C_i in simulation s_j. We further define C_i^{min} and C_i^{max} as the minimal and maximal values respectively for coverpoint C_i, across all simulations in S.* ■

Example 1. *Fig. 1 shows the intervals $[min(C_i^j) : max(C_i^j)]$ for three coverpoints, $i \in [1:3]$, monitored over 10 simulation runs, $j \in [1:10]$. For example, the coverpoint, C_1, represents the* range *of output voltage of an LDO. Fig. 1 shows the range of C_1 in each simulation in red. Due to varying choice of X in the 10 simulations, different values of C_1^j are reported.*

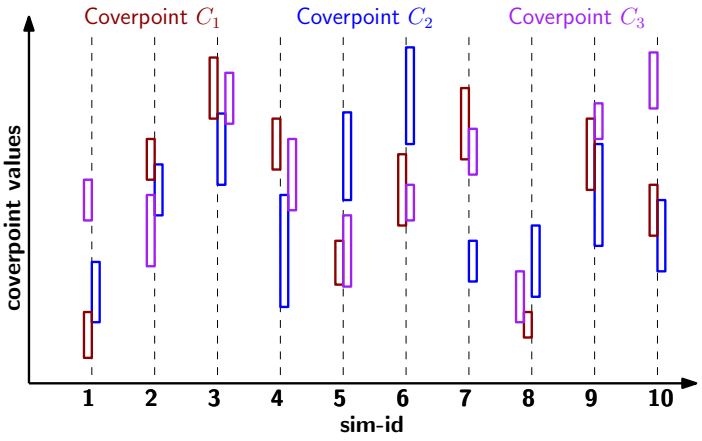

Fig. 1: A pictorial description of the selection problem

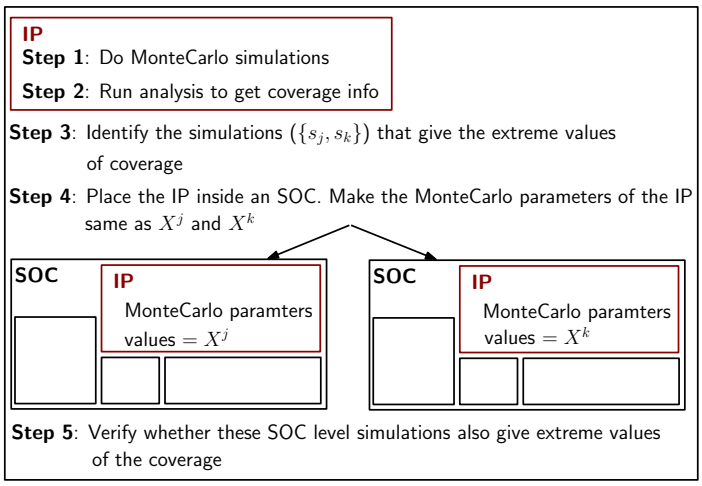

Fig. 2: A step-wise representation of the process

Definition 3. [Coverpoint Goal]*: The goal of a coverpoint, C_i, specifies the extremality of interest for that coverpoint, namely min, max, or both. Simulation s_j satisfies the coverpoint goal, g_i, if:*

- *g_i is min and we have $min(C_i^j) = C_i^{min}$, or*
- *g_i is max and we have $max(C_i^j) = C_i^{max}$, or*
- *g_i is both, $min(C_i^j) = C_i^{min}$, and $max(C_i^j) = C_i^{max}$.* ■

Our aim is to identify the smallest subset $S' \subset S$ that collectively meets all *coverpoint goals*, $g_i \; \forall i \in [1:m]$. The necessity of finding such a subset for a given design IP is as follows. The combination of parameters used in the runs of S' represent the corners at which the IP exhibits extremal behaviours with respect to the coverpoints of interest. When the IP is integrated in a design, we wish to exercise these corners to ensure that the IP does not exhibit any behaviour beyond these extremities. It is not practical to run Monte Carlo simulations at the higher levels of the design hierarchy due to the large number of corners and the enormous cost of running analog simulations. Therefore it is necessary to choose those corners that exercise the extremal behaviours of the component IPs. If

this can be enabled bottom-up, then the verification is focused on the relevant corners. Fig. 2 shows the steps in the verification flow.

In general, no single simulation will reach all the coverpoint goals, and therefore we need a cover, as illustrated by the following example.

Example 2. *In Fig. 1, if the coverpoint goal is min for each of C_1, C_2, and C_3, then the minimum cover is $\{s_1, s_8\}$. If the coverpoint goal for C_1 and C_2 is min and that for C_3 is max, then the minimum cover is $\{s_1, s_{10}\}$. If the coverpoint goals are min for C_1, max for C_2, and both for C_3, then the minimum cover is $\{s_1, s_6, s_8, s_{10}\}$.* ■

III. Complexity of Finding A Cover

We define the SIMCOV problem as follows. We are given a set, $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, of n simulations, and a set, $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$, of m coverpoints monitored in those runs. Let $I_{ij} = [a_{ij}, b_{ij}]$ denote the range of values for C_i covered in simulation, s_j. We are also given the coverpoint goal, $g_i \in \{min, max, both\}$, for each C_i. The objective is to find a minimum subset of $\mathcal{S}$ which covers all the coverpoint goals. The following theorem establishes that this problem is NP-hard.

Theorem 1. SIMCOV *is* NP-hard.

Proof: We establish a polynomial time reduction from the *set cover* problem, SETCOV. Given a set of subsets, $\mathbb{S} = \{S_1, S_2, S_3, \cdots, S_n\}$, of a set, $\mathcal{U}$, a minimum cover is a minimum subset of $\mathbb{S}$ whose union is $\mathcal{U}$.

For a given instance of the set cover problem, we create an instance of the SIMCOV problem in polynomial time, as follows. We define $\mathcal{U}$ as a set of coverpoint goals, that is, each element of $\mathcal{U}$ represents a coverpoint goal. Each subset, $S_i \in \mathbb{S}$, represents the set of coverpoint goals reached in simulation s_i. Seeking a minimum cover of $\mathbb{S}$ is therefore the problem of seeking a minimum set of simulations that reach all coverpoint goals. Therefore SETCOV $\preccurlyeq_P$ SIMCOV, and since SETCOV is known to be NP-hard, so is SIMCOV. ■

IV. Proposed Approach

In this section we present an approach based on Satisfiability Modulo Theory (SMT) and discuss progressive relaxation approaches for finding a cover of acceptable size.

A. The basic SMT formulation

We prepare the SMT formulation of SIMCOV with n Boolean variables, $s_1, s_2, \ldots, s_n$, where n is the number of simulations. The truth of variable s_i represents the choice of s_i in the cover, that is, $s_i = 1$ iff simulation S_i is part of the cover, else $s_i = 0$. Following are the basic SMT clauses.

- *Condition 1: All goals must be satisfied*
  For each coverpoint goal, g_p, we define a clause:

$$\mathbb{C}(g_p) : \bigvee \{s_i \mid \text{simulation } S_i \text{ reaches } g_p\}$$

  Since all coverpoint goals must be reached collectively, we add the following:

$$\mathbb{C}_1 : \bigwedge \mathbb{C}(g_p),\ \forall p \in [1:m]$$

Algorithm 1: Solving with SMT

Input: Set of values of coverpoints across all simulations, $C = \{C_p^q | \ \forall p \in [1:n], \forall q \in [1:m]\}$,
set of goal conditions for all m coverpoints, $G = \{g_1, g_2, g_3, \cdots, g_m\}$,
set of weights for all goal conditions $W = \{w_1,\ w_2,\ \cdots,\ w_m\}$,
size of the optimal set k $(< n)$
Output: Subset of simulation run of size k

```
1 verdict = unsat
2 while verdict == unsat do
      // Creating the SMT instance
3     SIMCOV = generate_smt_instance(n, m, k, C, G ,W)
      // Solving the SMT instance
4     verdict = SIMCOV.solve()
5     if verdict == unsat then
          // Find the unsat core
6         U_c = SIMCOV.unsat_core()
          // Find goal conditions in unsat core
7         G_U = {g_i | g_i ∈ U_c, ∀i ∈ [1, m]}
          // Relax the goal conditions
8         G = relax_goal_conditions(G, G_U)
  // Return the solution of SMT solver
9 return SIMCOV.solution
```

- *Condition 2: Choosing a set of at most $k(< n)$ simulations*
  In order to ensure that at most k simulations are chosen, we add the following constraint:

$$\mathbb{C}_2 : \sum_{i=1}^{n} s_i \leq k$$

In order to find the minimum cover, we iterate over k. There are various ways to search the domain of k, including binary search.

B. Relaxation Approaches

In a large design, the minimum cover may be prohibitively large and simulating all the corners indicated by the cover may not be feasible. In such cases, we need to relax the constraints defining the cover, so that a smaller and acceptable cover is obtained. Suppose we choose a feasible value of k and use the SMT formulation with that value of k to discover that no cover of size k or less exists. This section explains our approach in such cases.

Modern SMT solvers are equipped with the ability to find a minimal *unsatisfiable core* of an unsatisfiable SMT formulation. Each unsatisfiable core consists of a minimal subset of clauses that cannot be satisfied together. Resolving the unsatisfiable cores is the key to relaxation. Suppose clause $\mathbb{C}(g_i)$ belongs to an unsatisfiable core, and without loss of generality suppose the coverpoint goal, g_i, is *max*. This means, $\mathbb{C}(g_i)$ is a disjunction of simulation vectors that *reach the maximum value for coverpoint, g_i*. If we relax this, for example, to include simulation, s_k, that *reaches at least 90% of the maximum value for coverpoint, g_i*, then $\mathbb{C}(g_i)$ gets enlarged to $\mathbb{C}(g_i) \vee s_k$. Adding literals into clauses participating in an unsatisfiable cores makes it eventually satisfiable. We propose the following approaches:

- *Choosing goal conditions to relax.* We relax the goal conditions present in the unsatisfiable core and continue the search for a solution. If the core contains clauses for multiple

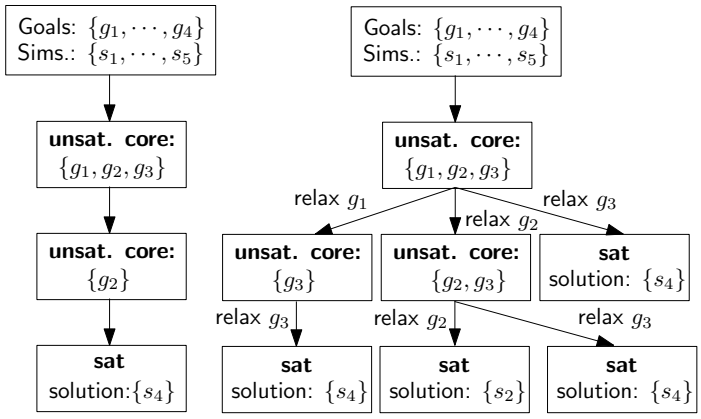

Fig. 3: Standard and randomised choice of relaxing *goal* conditions ($n = 5, m = 4, k = 1$)

coverpoints, we may relax the goal conditions for all the coverpoints, referred to as *standard relaxation*, or randomly choose a coverpoint whose goal condition is to be relaxed, *randomised relaxation*. In the latter option, we may also explore multiple alternatives, leading to multiple solutions that define a pareto-optimal front.

- *Percentage and Absolute relaxation.* The *goal relaxation criteria* may be of two types, namely, *percentage* and *absolute* relaxation. For example, consider a case where the set $\{1,\ 0.98,\ 0.93,\ 0.89,\ 0.83,\ 0.78\}$ represents the ordered set of values of a coverpoint as seen across all n simulations. The *goal condition* is to find all simulations that have hit the maximum value. A *percentage* relaxation, say 10%, to this *goal condition* translates to *finding all simulations where the value of coverpoint is* ≥ 0.9. In *absolute relaxation*, the next best value is considered. If *absolute relaxation* is used for the same example, the *goal condition* will be modified to *find all simulations where the value of coverpoint is* ≥ 0.98.
- *Attaching weights to goals:* Modern SMT solvers support the assignment of weights to each clause [14]. Depending on the criticality of the value of a coverpoint during the integration phase, weights may be attached to the respective *goal conditions* of these coverpoints.

Algorithm 1 contains the pseudo-code of the operations. Line 2-8 represents one epoch of the SMT instance. If unsatisfiable, the goal condition(s) present in the *unsat core* is(are) relaxed. Another instance of SMT is created with the modified goal conditions. Such epochs continue until a satisfiable SMT instance is found. Fig. 3 shows the two scenarios; *standard relaxation* where all *goal* conditions in the unsat core are relaxed at once, and *randomised relaxation* where a randomly selected goal condition of the unsat core is relaxed. Each arrow represents one epoch with the SMT solver. While standard relaxation follows a deterministic path and churns a single solution, s_4, randomised relaxation explores the search space non-deterministically and offers a set of solutions, $\{s_2, s_4\}$.

V. Experimental Results

We used Monte Carlo simulations on an industrial LDO circuit from Texas Instruments to find a suitable cover following the approach of Section IV-A. We have used Z3 solver [15] as our SMT solver. The Monte Carlo simulations resulted in $n = 1300$ simulation runs.

A total of $m = 6$ coverpoints were declared for computing the steady-state values of some critical nets, namely, `ldo_1v8`, `ldo_3v0`, `supp_5v0`, `vbg`, `vref_0p6`, and `vref_1p0`. The coverage results from the 1300 simulations were collected.

At first, we aimed to find a cover with a single simulation, that is, $k = 1$, where all coverpoint goals are *max*, and another cover where all coverpoint goals are *min*. Solving the basic SMT formulation, we found that none of these covers exist. Therefore we need to use relaxation.

1) Table I shows the results from our approach when searching for the corner with coverpoint goals as *max*. Each row corresponds to a different configuration, as indicated in Section IV-A.
   - In Case #1 all the goal conditions present in the *unsat core* were relaxed at once, that is, *absolute relaxation* was followed, and all coverpoints carried equal weightage. After multiple epochs of relaxation, the solver reported sim-id 977 as the solution with $k = 1$. In a similar way, the solutions for $k = 2$ and $k = 3$ were found to be respectively the set of sim-id's $\{283, 971\}$ and $\{162, 282, 971\}$.
   - Case #2 is similar to Case #1 except that we associate weights with coverpoints, thereby associating a preference on relaxing coverpoint goals. Since the nets `ldo_1v8` and `ldo_3v0` were comparatively more critical than the others in our application, higher weights were assigned to the coverpoints associated with these nets. With this modified preference, the solutions for $k = 2$ and $k = 3$ were found to be respectively the set of sim-id's $\{374\}$, sim-id's $\{283, 966\}$, and sim-id's $\{283, 595, 974\}$.
   - For cases where the *goal conditions* were selected randomly from the *unsat core*, the solver reported different solutions in each iteration. With a 5% relaxation, six solutions were reported for $k = 1$ (Case #3) with the weighted goal condition. Similarly, three and two covers were reported for $k = 2$, and $k = 3$ respectively.
2) Table II reports the number of epochs, the time taken, and the relaxation on each goal for arriving at the solutions of Table I. For cases where *standard relaxation* was followed, the relaxation values are integers representing the rank of the value of the coverpoint at the last epoch. For example, a relaxation by 3 for `vref_0p6` in the first row signifies that the coverpoint goal was relaxed 3 times and thus the goal value at the last epoch is the 4^{th} highest value of the coverpoint among all 1300 simulations.
3) Table IV shows the maximum values of each coverpoint across 1300 simulations. It also reports the maximum values of the coverpoints in the simulations identified in Table I. According to our requirements, we selected one solution from Case#5, sim-id 972, as our simulation corner for getting maximum values for all coverages. For the corner with minimum coverage values, sim-id 417 was chosen with

Case #	Configuration			Result (sim-id's)		
	type of relaxation	relaxation criteria	weights on goal condition	$k=1$	$k=2$	$k=3$
1	standard	absolute	no weights attached	977	{283, 965}	{162, 282, 971}
2	,,	,,	increased weight for ldo_1v8, ldo_3v0	374	{283, 1242}	{283, 595, 972}
3	randomised	percentage, 5%	,,	361, 363, 537, 792, 1224, 1245	{361, 622}, {363, 971}, {621, 971}	{361, 623, 971}, {360, 595, 971}
4	,,	percentage, 0.05%	equal weights	101, 1245	{283, 972}, {283, 974}, {283, 968}	{282, 595, 971}, {283, 595, 971},
5	,,	,,	increased weight for ldo_1v8, ldo_3v0	972, 1224	{283, 974}, {283, 966}, {283, 965}	{283, 597, 972}, {164, 282, 971}

TABLE I: Results on LDO coverage results $(n = 1300,\ m = 6)$

Case	Cover	No. of epochs	Relaxation	Time Taken(s)
1	977	38	{28, 28, 0, 29, 3, 10}	156.6
	{283, 965}	9	{5, 8, 0, 6, 2, 5}	17.6
	{162, 282, 971}	4	{3, 3, 0, 3, 1, 3}	6.8
2	374	42	{41, 27, 0, 19, 3, 9}	164.3
	{283,1242}	10	{7, 8, 0, 8, 3, 8}	29.2
	{283, 595, 972}	4	{3, 3, 0, 3, 1, 3}	11.8
3	361	4	{0.05, 0.05, 0, 0.05, 0, 0}	463.2
	363	4	{0.05, 0.05, 0, 0.05, 0, 0}	363.2
	537	10	{0.05, 0.25, 0, 0.05, 0.05, 0.05}	1017.2
	792	5	{0.05, 0.05, 0, 0.05, 0, 0.05}	436.4
	1224	7	{0.05, 0.1, 0., 0.05, 0.05, 0.05}	959.1
	1245	8	{0.05, 0.15, 0, 0.05, 0.05, 0.05}	1002.1
	{361, 622}	3	{0.05, 0.05, 0, 0, 0, 0}	185.5
	{363, 971}	3	{0, 0.05, 0, 0.05, 0, 0}	487.6
	{621, 971}	4	{0, 0.05, 0, 0, 0.05, 0.05}	515.8
	{361, 623, 971}	2	{0, 0.05, 0, 0, 0, 0}	488.7
	{360, 595, 971}	2	{0, 0, 0, 0.05, 0, 0}	364.7
4	101	131	{0.0155, 0.0195, 0, 0.0185, 0.004, 0.0085}	273.2
	1245	128	{0.015, 0.018, 0, 0.018, 0.0035, 0.009}	222.6
	{283, 972}	70	{0.0065, 0.0065, 0, 0.01, 0.005, 0.0065}	83.1
	{283, 974}	48	{0.003, 0.0065, 0, 0.007, 0.0025, 0.0045}	72.2
	{283, 968}	49	{0.003, 0.0065, 0, 0.0075, 0.0035, 0.0035}	74.4
	{282, 595, 971}	37	{0.006, 0.0035, 0, 0.0035, 0.002, 0.003}	53.1
	{283, 595, 971}	22	{0.0005, 0.0025, 0, 0.0025, 0.002, 0.003}	26.3
5	972	111	{0.0155, 0.014 , 0, 0.0175, 0.0035, 0.006}	166.0
	1224	111	{0.0175, 0.0125, 0, 0.0165, 0.0035, 0.006}	169.5
	{283, 974}	50	{0.004, 0.0065, 0, 0.007, 0.0035, 0.0075}	80.5
	{283, 966}	54	{0.0045, 0.0065, 0, 0.0055, 0.0035, 0.0065}	79.4
	{283, 965}	65	{0.005, 0.0065, 0, 0.011, 0.0035, 0.006}	117.8
	{283, 597, 972}	27	{0.0025, 0.002 , 0, 0.0035, 0.002, 0.003}	35.2
	{164, 282, 971}	50	{0.009, 0.005, 0, 0.0045, 0.002, 0.004}	114.2

TABLE II: Time taken and goal relaxation incurred in reaching each solution. Relaxation values for goals given in the order of {`ldo_1v8,ldo_3v0,supp_5v0,vbg,vref_0p6,vref_1p0`}.

Category	signal	min. value (V)	max. value (V)
IP level simulation			
IP level Monte Carlo of 1300 simulations	ldo_1v8	1.7205	1.8715
	ldo_3v0	2.3985	3.0625
Module level simulation			
Module level corners simulation (Base line)	ldo_1v8	1.79	1.81
Nominal, weak, strong, skewn, skewp process	ldo_3v0	2.4	3.001
Module level corner simulation with LDO block level	ldo_1v8	1.73	1.86
MC parameters from sim: 417 and 972	ldo_3v0	2.4	3.016

TABLE III: Hierarchical coverage analysis of LDO circuit

net	ldo_1v8	ldo_3v0	supp_5v0	vbg	vref_0p6	vref_1p0
max. values in 1300 sims.	1.8715	3.0625	5.4995	1.2435	0.6045	1.0105
sim-id 101	1.8435	3.0095	5.4995	1.2265	0.6035	1.0065
sim-id 162	1.7855	3.0485	5.4995	1.2085	0.5995	1.0015
sim-id 164	1.7855	3.0485	5.4995	1.2075	0.5995	1.0015
sim-id 282	1.7935	2.3985	2.4005	1.2495	0.6025	1.0085
sim-id 283	1.7935	3.0435	5.4995	1.2495	0.6025	1.0085
sim-id 360	1.8055	2.3985	2.4005	1.2155	0.6045	1.0105
sim-id 361	1.8065	2.9965	5.4995	1.2155	0.6045	1.0105
sim-id 363	1.8035	2.9955	5.4995	1.2155	0.6045	1.0105
sim-id 374	1.8105	3.0315	5.4995	1.2405	0.6015	1.0045
sim-id 537	1.8085	2.9835	5.4995	1.2195	0.6015	1.0015
sim-id 595	1.8365	3.0625	5.4995	1.1565	0.6015	1.0045
sim-id 597	1.8345	3.0595	5.4995	1.1555	0.6015	1.0045
sim-id 621	1.7985	3.0035	5.4995	1.2495	0.6015	1.0005
sim-id 622	1.7975	2.3985	2.4005	1.2495	0.6015	0.9995
sim-id 623	1.7975	3.0025	5.4995	1.2495	0.6015	0.9995
sim-id 792	1.7895	3.0255	5.4995	1.1955	0.6045	1.0045
sim-id 965	1.8625	2.3995	2.4005	1.1855	0.5995	0.9965
sim-id 966	1.8635	3.0015	5.4995	1.1855	0.5995	0.9965
sim-id 968	1.8665	3.0075	5.4995	1.1885	0.6015	0.9995
sim-id 971	1.8715	2.3985	2.4005	1.1935	0.6035	1.0045
sim-id 972	1.8705	3.0205	5.4995	1.1935	0.6035	1.0045
sim-id 974	1.8705	3.0195	5.4995	1.1925	0.6035	1.0045
sim-id 977	1.8295	3.0225	5.4995	1.2275	0.6015	1.0025
sim-id 1224	1.8395	3.0245	5.4995	1.2095	0.5995	1.0015
sim-id 1242	1.8555	2.3985	2.4005	1.2255	0.6005	0.9985
sim-id 1245	1.8545	3.0175	5.4995	1.2295	0.6025	1.0015

TABLE IV: Maximum values of coverages across different simulation runs of the LDO as identified in Table I

a similar configuration and in a similar fashion.

Tables I and IV reveal the trade-off between the size of the cover and the degree of relaxation on the coverage extremalities to obtain a cover of that size. A comparison of the solutions of Case-1 in Table I shows that restricting the value of k (maximum number of simulations present in the solution cover) to 1 limits the coverage of the extremalities. With $k = 2$, the solver is able to find simulation ids containing better coverage of the coverpoints. As compared to sim-id 977, better values for `ldo_3v0`, `vbg`, and `vref_1p0` were reported in sim-id 283, while sim-id 965 improved the coverage values for `ldo_1v8`. Similar conclusions can be drawn from the other cases. In practice, the ability to find such pareto-optimal alternatives is essential for balancing verification time with verification coverage.

Table III illustrates another utility of our approach. The second row shows the ranges of `ldo_1v8` and `ldo_3v0` at IP level. Once the LDO IP gets integrated with other IPs, the module-level simulations across the process corners report range of $[1.79 : 1.81]$ for `ldo_1v8`. This reveals coverage gaps of $[1.7205 : 1.79]$ and $[1.81 : 1.8715]$. On carrying out two module-level simulations with the LDO containing Monte Carlo parameters from sim-id 417 and 972, the range gets enhanced to $[1.73 : 1.86]$, closing the coverage gap to some extent. Similar outcomes can also be noticed for `ldo_3v0`.

VI. Conclusion

The task of choosing the simulation parameters for covering extremal behaviors of AMS design components is a complex one when dealing with a design having multiple analog components. We provide a formal basis for choosing the parameter combinations by leveraging the coverage information from component level Monte Carlo simulations, extracting the parameter combinations with which extremal behaviors were experienced, and then finding the minimum set of combinations that cover all the extremalities. We believe that this is a useful information in practice and our experiments show that this helps in narrowing the coverage gap between IP-level simulations in isolation, and in context when the IP is integrated in a design. The proposed approach enables a hierarchical analytical framework for coverage driven selection of simulation corners in AMS designs.

References

[1] R. Saleh, S. Wilton, S. Mirabbasi, A. Hu, M. Greenstreet, G. Lemieux, P. Pande, C. Grecu, and A. Ivanov, "System-on-Chip: Reuse and Integration," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1050–1069, 2006.

[2] J.-M. Jou, S.-R. Kuang, and K.-M. Wu, "A hierarchical interface design methodology and models for SoC IP integration," in *2002 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, 2002, pp. II–II.

[3] T. M. Hancock and J. C. Demmin, "Heterogeneous and 3D Integration at DARPA," in *2019 International 3D Systems Integration Conference (3DIC)*, 2019, pp. 1–4.

[4] H. Hu, P. Li, and J. Z. Huang, "Enabling High-Dimensional Bayesian Optimization for Efficient Failure Detection of Analog and Mixed-Signal Circuits," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19, Las Vegas, NV, USA: Association for Computing Machinery, 2019.

[5] H. Hu, P. Li, and J. Z. Huang, "Parallelizable bayesian optimization for analog and mixed-signal rare failure detection with high coverage," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '18, San Diego, California: Association for Computing Machinery, 2018.

[6] M. Ma, L. Hedrich, and C. Sporrer, "A machine-readable specification of analog circuits for integration into a validation flow," in *FDL 2011 Proceedings*, 2011, pp. 1–8.

[7] "IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows," *IEEE Std 1685-2018 (Revision of IEEE Std 1685-2014)*, pp. 1–510, 2018.

[8] W. Kruijtzer, P. van der Wolf, E. de Kock, J. Stuyt, W. Ecker, A. Mayer, S. Hustin, C. Amerijckx, S. de Paoli, and E. Vaumorin, "Industrial IP integration flows based on IP-XACT standards," in *2008 Design, Automation and Test in Europe*, 2008, pp. 32–37.

[9] E. Pekkarinen, M. Teuho, E. Salminen, and T. D. Hamalainen, "Resolving parameter reference management in IP-XACT using Kactus2," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 002 765–002 770.

[10] C. Z. Mooney, *Monte carlo simulation*, 116. Sage, 1997.

[11] S. Sanyal, A. Hazra, P. Dasgupta, S. Morrison, S. Surendran, and L. Balasubramanian, "CoveRT: A Coverage Reporting Tool for Analog Mixed-Signal Designs," in *33rd International Conference on VLSI Design*, 2020, pp. 119–124.

[12] S. Sanyal, A. Hazra, P. Dasgupta, S. Morrison, S. Surendran, and L. Balasubramanian, "The Notion of Cross Coverage in AMS Design Verification," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 217–222.

[13] S. Sanyal, P. Dasgupta, A. Hazra, S. Das, S. Morrison, S. Surendran, and L. Balasubramanian, "The CoveRT Approach for Coverage Management in Analog and Mixed Signal Integrated Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[14] F. Heras, J. Larrosa, and A. Oliveras, "Minimaxsat: An efficient weighted max-sat solver," *Journal of Artificial Intelligence Research*, vol. 31, pp. 1–32, 2008.

[15] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340.