Functional Coverage-Driven Characterization of RF Amplifiers

Muhammad Hassan[1,2], Daniel Große[1,2], Thilo Vörtler[3], Karsten Einwich[3], Rolf Drechsler[1,2]
[1]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
[2]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[3]COSEDA Technologies GmbH, Dresden, Germany
muhammad.hassan@dfki.de {grosse,drechsle}@informatik.uni-bremen.de
{thilo.voertler,karsten.einwich}@coseda-tech.com

***Abstract*—In this paper we propose the first functional coverage-driven characterization approach as a systematic solution for the class of *Radio Frequency* (RF) amplifiers. We elevate the main concepts of digital functional coverage to the context of SystemC AMS in particular, and system-level simulations in general. To enable AMS functional coverage-driven characterization, we introduce two coverage refinement parameters on input and output side, to systematically generate input stimuli and capture specifications. At the heart of the approach is the coverage analysis which measures the functional coverage of the DUV and provides clear feedback to reach coverage closure. We provide a case study using an industrial RF transmitter and receiver model to demonstrate the applicability and efficacy of our approach.**

***Index Terms*—SystemC AMS, Virtual Prototyping, Functional Coverage, Analog/Mixed Signal, RF Amplifiers**

I. Introduction

Industry 4.0, the forefront for *Internet-of-Things* (IoT), and *Cyber Physical Systems* (CPS), has significantly altered the requirements for *Analog Mixed Signal* (AMS) *System-on-Chips* (SoC). Tight integration of analog and digital *Intellectual Properties* (IPs) on a single die, while running software on top has significantly increased the functionality and reduced the area of the SoC. However, the design complexity has increased manyfold making the verification of AMS SoCs an increasingly daunting task. The major reasons are 1) the continuous time and continuous value behavior of the analog signals and their complex dependencies, enlarging the number of possible scenarios to infinity, 2) slow *Simulation Program with Integrated Circuit Emphasis* (SPICE) level simulations [1], [2], [3], 3) design specific input signals, and 4) often manual observation of the *Design Under Verifiaction* (DUV) output.

Fortunately, the abstraction of SystemC AMS *Virtual Prototypes* (VPs) offer a good trade-off between design accuracy and simulation speed [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. The early availability, support for SystemVerilog-like assertions/checkers [14], [15], and significantly faster simulation speed as opposed to SPICE simulations [16] allows these models to be used as a reference for functional verification of the SoC at lower abstractions, i.e., transistor level. Hence, their functional correctness is inevitable.

In digital design the verification quality is ensured by tracking the verification progress and completeness. Coverage is the metric used for this task, in particular functional coverage since it allows to measure if all features of the design have been verified [17]. While functional coverage is very well understood in digital design (see e.g. [18], [19], [20]), this is not the case for AMS [21], [22]; in related work section we discuss this in more detail. As a consequence, the existing AMS-verification approaches are not coverage-driven and hence not systematic [23]. Due to the rising complexity of AMS designs this becomes a serious problem as corner cases may be missed or even general features are not thoroughly verified.

Contribution: In this paper, we propose the first functional coverage-driven characterization approach as a systematic solution for *Radio Frequency* (RF) amplifiers (*Power Amplifiers (PAs), Low Noise Amplifiers (LNAs), Driver Amplifiers (DAs)* etc). We elevate the main concepts of digital functional coverage to the context of SystemC AMS in particular and system level simulations in general.

First, to enable AMS functional coverage-driven characterization, we introduce two coverage refinement parameters on input and output side, to systematically define the input stimuli and capture the DUV specifications. More precisely, on the input side we define the static parameters[1] of the input stimuli signals using the refinement parameters, i.e., range and input resolution. On the output side, the refinement parameters are used to define the functional coverage to capture the DUV specifications.

Second, we present a complete functional-coverage driven characterization approach. At the heart of the approach is the coverage analysis which uses the functional coverage of input, output, and checkers, to determine whether all DUV features according to the specification have been verified. In case of uncovered features, it provides hints to revise the refinement parameters to increase coverage, hence, eventually fully characterizing the DUV.

We use an industrial RF transmitter and receiver model as

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project CONVERS under contract no. 16ES0656, and University of Bremens graduate school SyDe, funded by the German Excellence Initiative.

[1]Static parameters are the parameters which remain constant during the execution of one stimuli signal, e.g., amplitude, frequency, phase etc.

a case study to demonstrate the applicability and efficacy of our approach.

II. Related Work

Coverage in the digital domain is a well researched topic, see for instance [18], [24], [17], [19], [25], [26]. Different coverage metrics for digital circuits have been developed. Essentially, they range from structural coverage, e.g. code coverage (find unexercised lines of code), to functional coverage (find unexercised functionality based on user-defined scenarios to be expected). While the structural coverage metrics are very important in the beginning of verification, functional coverage is the metric of choice when ensuring the overall verification quality as the checks are performed wrt. the specification [17].

The application of coverage metrics in AMS design, however, is still in its infancy [22]. A major reason is that digital metrics have to consider discrete values only. In contrast, in analog we have to deal with continuous signals. In our particular case we benefit here from the SystemC AMS abstraction.

First analog metrics have been inspired from the test community, e.g. [27], and only target transistor level abstraction. The authors of [28], [22], [29] proposed several novel coverage metrics for AMS circuits like fault coverage, parameter coverage, state space coverage, data flow coverage, and test case coverage. While the definition of metrics is an important step, functional coverage in our sense is not considered, and a coverage-driven characterization approach is missing.

The authors of [30], [31] propose a feature indented assertion language, and a methodology to accelerate the feature coverage by learning the map between input space, and feature space, and using this learning map to generate input stimuli to achieve coverage closure. However, these works only target transistor level designs.

III. Preliminaries and Problem

In this section we briefly review functional coverage. Then, we introduce our running example, a *Low Noise Amplifier* (LNA). The last part of this section reviews the typical (advanced industrial) verification environment at the system-level and identifies its deficiencies.

A. Functional Coverage

Functional coverage is a verification metric heavily used in digital verification [17]. It determines the extent to which the functionality (or features) of the DUV have been exercised by the input stimuli. Functional coverage is defined by the verification engineer in accordance with the specifications of the DUV. According to the IEEE SystemVerilog standard [14], the following ingredients are used in a simulation-based verification setting. A coverage model is required to define functional coverage, defined as *covergroup*. Each *covergroup* can contain one or more *coverpoints*. Functional coverage maps each functional aspect (or specification) to a *coverpoint*. Each *coverpoint* contains *coverage bins* (sometimes referred to as *bins*) which collect and calculate the number of occurrences of various values. Functional coverage also allows to track information which is received simultaneously on multiple *coverpoints*, called *cross-coverage*. One of the advantages of functional coverage is that it can be reused for verification at different design abstractions.

B. Running Example: LNA

As running example we use a real-world model of an LNA, i.e. the behavioral model has been taken from [32]. In general, an LNA amplifies a weak low power input signal without affecting its *Signal-to-Noise-Ratio* (SNR) significantly. It is used in various applications in RF front ends, e.g., mobile phones, automotive keyless entry devices, *Wireless LANs* (WLAN) etc. The concrete LNA has the following specifications:

- *Gain* (G) (min., typical, max.) = 16.5 dB, 18.2 dB, 19.9 dB
- 1dB compression point = 30 dBm
- Output *Third-Order Intercept* (IP3) = 70 dBm
- Operating frequency = 5 KHz to 20 KHz
- Input impedance = 50 Ohms
- Output impedance = 50 Ohms

The model is implemented in SystemC AMS.

C. AMS Verification Environment and Deficiencies

To verify an AMS DUV a verification environment has to be created. Fig. 1 shows such a verification environment surrounding our running LNA example. A general verification environment consists of the light blue elements in Fig. 1: A signal generator on the input side, a DUV, and assertions/checkers on the output side. The signal generator is used for generating input stimuli for the DUV and the assertions/checkers are placed to check correctness of functionality. If the checkers pass, the DUV behavior is correct.

Based on this verification environment the verification engineer performs the characterization of the LNA. For thorough characterization of the LNA however, a systematic approach as motivated in the introduction is necessary which allows to control (and check) the stimuli side as well as to check whether all design features have been characterized. Hence, we extend the verification environment by several components as shown in the gray area. They form the basis for our proposed AMS functional coverage-driven characterization approach as detailed in the next section.

IV. AMS Functional Coverage-Driven Characterization Approach

In this section we present the proposed AMS functional coverage-driven characterization approach. At first, we provide a general overview of the approach which includes a brief explanation of the main ingredients: input stimuli generation, output coverage definition, cross-coverage definition, checkers definition, and coverage analysis. Afterwards, we detail all of them while always providing concrete examples illustrating each aspect for our LNA running example.

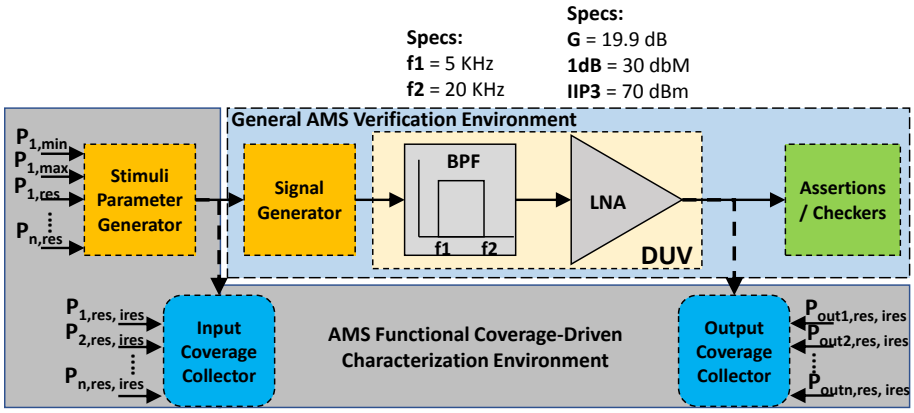

Fig. 1. AMS functional coverage-driven characterization Environment - LNA: Low Noise Amplifier, BPF: Band-Pass Filter, res: resolution, ires: interval resolution

A. Approach Overview

The overall approach for the characterization of RF amplifiers (LNAs in particular) driven by AMS functional coverage is shown in Fig. 2. The approach consists of five systematic steps, 1) input stimuli generation, 2) output coverage definition and collection, 3) cross-coverage definition, 4) checkers definition, 5) and coverage analysis. Input stimuli generation comprises of the following three stages, stimuli static parameter generation, input coverage definition and collection, and signal generation. Input stimuli static parameters are generated and stored in *parameters database*. Different static parameters configure the *signal generator* to generate different input stimuli signals for the DUV, leading the stimulus to exercise different functionality. This can be observed using the AMS functional coverage coverpoints on input and ouput, i.e., *input coverage collector*, and *output coverage collector*. A *cross-coverage* is defined between *input coverage collector* and *output coverage collector* to examine the relationship between inputs and outputs. The *checkers* are used to ensure the correctness of the DUV output behavior. When all the parameters from the *parameters database* have been used and the database is empty, *coverage analysis* is done. The *coverage analysis* systematically guides the verification engineer to revise the stimulus static parameters, *input coverage collector*, or *output coverage collector* to maximize functional coverage. The overall goal of the proposed approach is to finally achieve 100% functional coverage, ideally.

In the following we detail the ingredients of our AMS functional coverage-driven characterization approach for RF amplifiers as well as demonstrate them using the running example.

B. Input Stimuli Generation and Coverage

The first step for AMS functional coverage-driven characterization is the generation of input stimuli signals. They are generated using an ideal signal generator, which is defined as a function

$$f(t, p1, p2, p3, ...pi)$$

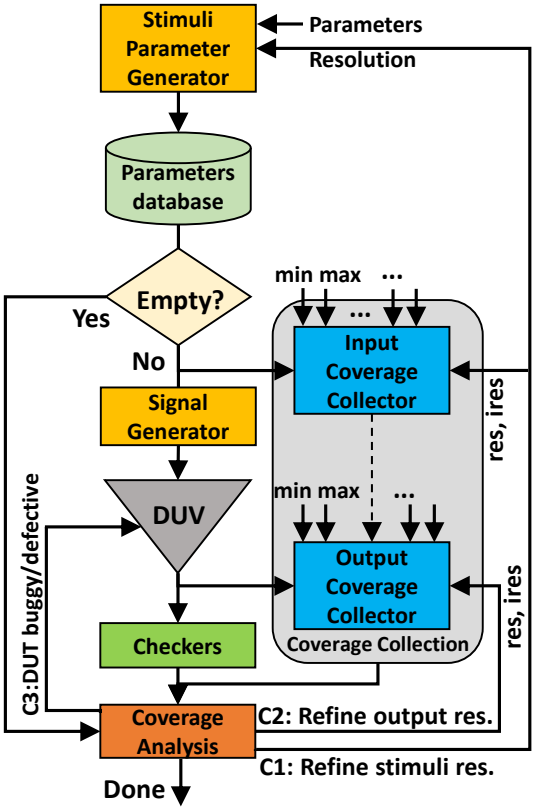

Fig. 2. Overview on AMS functional coverage-driven approach – res: resolution, ires: interval resolution

where t is time, and $p1, p2, ..., pi; 0 < i < \infty$ are the stimuli static parameters which shape the input stimuli signals. The selection of function $f(t, ...)$ is based on the DUV functionality to verify, e.g., sine, square wave, single-tone, multi-tone, user defined etc.

Input stimuli generation comprises of three stages, input stimuli static parameter generation, input coverage collection, and signal generation. In the following, these three stages are discussed in detail.

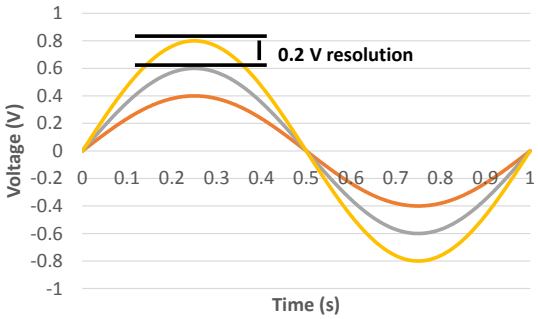

Fig. 3. Stimuli parameter resolution definition with 0.2 V amplitude difference

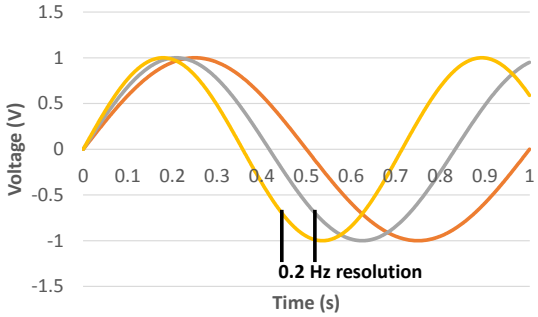

Fig. 4. Stimuli parameter resolution definition with 0.2 Hz frequency difference

1) Stimuli Static Parameter Generation: Stimuli static parameters are defined after consulting the specifications of the DUV from the datasheet. The datasheet defines the possible range of input stimuli static parameters, e.g., minimum/maximum frequency (Hz), minimum/maximum amplitude (V), input power range (dBm) etc.. Furthermore, an efficient parameter sweeping is required in the defined range, as it is not feasible to simulate for each possible value. Therefore, a *stimuli parameter generator* (Fig. 2) is defined which takes as input a range of the stimuli parameters and an input *resolution* parameter, i.e., the step-size, to systematically generate static parameters. As an example, Fig. 3 shows sine waves with *resolution* 0.2 V amplitude difference and Fig. 4 shows sine waves with 0.2 Hz frequency difference. This input *resolution* definition is important and its selection for stimuli parameters should be carefully done. Too coarse *resolution* will result in having few stimuli static parameters, leading to few input stimuli signals. On the other hand, too fine *resolution* will result in too many input stimuli signals. In case of former, the simulation time is reduced at the expense of possibly unverified DUV. Whereas, for latter, the simulation time is extremely high and possibly a fully verified DUV. Please note, coarse *resolution* means a bigger step-size, and a fine *resolution* means a smaller step-size. The *stimuli parameter generator* generates all possible combinations of static parameters w.r.t. the defined *resolution* and stores them in the *parameters database*.

For our running example of LNA, we take an ideal signal generator which takes two arguments as input, and has the following function implemented

$$f_1(t, A, F) = A * sin(2 \times \pi \times F \times t) \tag{1}$$

where parameter *A* is amplitude, parameter *F* is frequency, and parameter *t* is time of the stimuli signal. *A* and *F* are static parameters which need to be defined. Hence, from the specifications of (Section III-B), initially we set the inputs of the stimuli parameter generator as follows

$$\begin{aligned} A &= (0, 5.0; 0.5)\ V \\ F &= (5e3, 20e3; 3e3)\ Hz \end{aligned} \tag{2}$$

Equation 2 specifies parameter *A*: between 0 V and 5 V (both inclusive) and *resolution* is 0.5 V. It means the amplitude values should be 0 V, 0.5 V, 1 V,... etc. Equation 2 also specifies parameter *F*: between 5 KHz and 20 KHz (both inclusive) and *resolution* is 3 KHz. It means the frequency values should be 5 KHz, 8 KHz, ... etc. The *stimuli parameter generator* takes these two parameters as inputs and generates pairs of parameters *(a, f)*. The concrete pairs are: *(a, f) = 0, 5e3; 0.5, 5e3; 1, 5e3;...; 4, 20e3*, ... etc.

2) Input Coverage Collection: Functional coverage coverpoints are required on the input side of the DUV to assess the quality of the generated static parameters. Hence, an *input coverage collector* is defined and placed before the *signal generator*. The reason for this placement is that the static parameters are directly available. Otherwise, if placement is done after *signal generator* then complex measurement tools and post-processing of the signal is required to extract the same parameters. The *input coverage collector* captures different stimuli static parameters, e.g., frequency (Hz), amplitude (V) etc. Additionally, it captures input specifications of the DUV, e.g., input power (dBm). The functional coverage *bins* for static parameters and input specifications are defined using the range of static parameters similar to Section IV-B1. Additionally, each *bin* defines another parameter *interval resolution*. Its purpose is to create a range of values inside the defined *bin resolution*. Please note, it is quite possible that the selected *resolution* and *interval resolution* results in *coverage bins* which do not cover all DUV input specifications. In this case a revision of *resolution* and *interval resolution* is required. The *input coverage collector* plays a vital role in cross coverage, explained later.

A relevant code snippet of the running example Section III-B for input coverage *coverpoints* is shown in Fig. 5. In the input coverage collector, *coverage bins* are defined for *A* (Fig. 5 Line 1 - Line 7), and *F* (Fig. 5 Line 9 - Line 15) using initially the same *resolution* as defined for stimuli static parameters in Equation 2. Hence, each *bin* represents one amplitude/frequency value. One additional *coverpoint* is defined for input power as follows

$$\begin{aligned} input\_power &= (20, 30; 5, 4.99)\ dBm \\ input\_power &= (30, 36; 2, 1.99)\ dBm \\ input\_power &= (38, 40; 1, 0.99)\ dBm \end{aligned} \tag{3}$$

TABLE I
LNA: PARAMETER A (AMPLITUDE) COVERAGE REPORT

	amplitude_cvp "amplitude_values"			
	Description	**Value**	**#Hits**	**Hit**
100.0%	low corner	0	18	✓
	0.5 V	0.5	18	✓
	1.0 V	1.0	18	✓
	1.5 V	1.5	18	✓
	2.0 V	2.0	18	✓
	2.5 V	2.5	18	✓
	3.0 V	3.0	18	✓
	3.5 V	3.5	18	✓
	4.0 V	4.0	18	✓
	4.5 V	4.5	18	✓
	high corner	5.0	18	✓

resolution = 0.5 V

TABLE II
LNA: PARAMETER F (FREQUENCY) COVERAGE REPORT

	freq_cvp "frequency_values"			
	Description	**Value**	**#Hits**	**Hit**
100.0%	low corner	5000	33	✓
	8 KHz	8000	33	✓
	11 KHz	11000	33	✓
	14 KHz	14000	33	✓
	17 KHz	17000	33	✓
	high corner	20000	18	✓

resolution = 3 KHz

Equation 3 defines coverage *bins* for input power with three different *resolution* and *interval resolution* values, 1) between 20 dBm to 30 dBm with 5 dBm *resolution* and 4.99 dBm *interval resolution*, 2) between 30 dBm to 36 dBm with 2 dBm *resolution* and 1.99 dBm *interval resolution*, 3) and between 38 dBm to 40 dBm with 1 dBm *resolution* and 0.99 dBm *interval resolution*. This is because of the logarithmic scale for dBm where the increase in amplitude (A) of the stimuli signals results in smaller increases in dBm. Hence, these three different values are used to cover maximum behavior of the DUV. The *bins* are defined from Fig. 5 Line 18 to Fig. 5 Line 26. The defined *bins* for *A, F, and input power* are shown in Table I, Table II, and Table III, respectively. The tables can be interpreted in the following way, the first column shows the overall coverage of the *coverpoint*, second column shows the description of the *coverage bins*, third column shows the value of each *bin* which should appear on the output of DUV for this *bin* to be covered, fourth column shows the number of times this *bin* was covered (hit), and last column shows if the *bin* was covered (hit) or not. *Green* color reflects a successful hit and *red* color reflects a miss.

TABLE III
LNA: INPUT POWER COVERAGE REPORT

	input_power_cvp "input_power_values"			
	Description	**Value**	**#Hits**	**Hit**
100.0%	20 dBm	[20: 24.99]	18	✓
	25 dBm	[25: 29.99]	18	✓
	30 dBm	[30: 31.99]	18	✓
	32 dBm	[32: 33.99]	18	✓
	34 dBm	[34: 35.99]	18	✓
	36 dBm	[36: 37.99]	18	✓
	38 dBm	[38: 38.99]	18	✓
	39 dBm	[39: 39.99]	18	✓
	40 dBm	[40: 40.99]	18	✓

resolution = 5, 2, 1 (dBm), *interval resolution* = 4.99, 1.99, 0.99 (dBm)

3) Signal Generation: The stimuli static parameters stored in the *parameter database* (Section IV-B1) are taken out one pair at a time and given as input to the *signal generator*. The *signal generator* generates the corresponding test input signal for the DUV. When the stimuli parameters are applied, we observe that the input coverage is 100% for amplitude A, frequency F, and input power (dBm) as shown in Table I, Table II, and Table III, respectively.

C. Output Coverage Definition and Collection

The input stimuli signals generated by *signal generator* exercise different DUV behavior. The output of DUV goes to *output coverage collector*. The *output coverage collector* is defined to capture different DUV specifications, i.e., output signal power (dBm), gain (dB), 1dB compression point etc. Again, the *resolution* of output coverage collector is defined. Too coarse (low *resolution*) may miss important specifications, creating coverage holes. Too fine (high *resolution*) may lead to too many unnecessary values exhibiting similar behavior. Furthermore, *interval resolution* similar to Section IV-B2 is also defined. The bins with more hits can further be expanded to observe exact behavior.

The output specifications of interest for our running example of LNA DUV are gain and 1 dB compression point. We define a coverpoint for gain with *bins* ranging from 16.5 dB to 20 dB, with a *resolution* of 0.5 dB, and *interval resolution* of 0.2 dB. The 1 dB compression point lies at the input power corresponding to 18.9 dB gain. The coverage bins are shown in Table IV.

D. Cross-Coverage Definition

Cross-coverage is required to observe the input-output relationship of the DUV. Using this information, it is exactly known which input stimuli exercised what output behavior. Table VII shows a snippet of the cross-coverage between checkers - input power - gain.

E. Checkers Definition

Functional coverage only tells how much functionality of the DUV has been exercised. In order to verify if the covered functionality also exhibited correct behavior, assertions or checkers are required. Hence, DUV output is also used as input for checkers as shown in Fig. 2.

F. Coverage Analysis

The coverage analysis is executed when all stimuli signals corresponding to static parameters have been generated and the *parameters database* is empty. The generated coverage results are analyzed to determine if all input *coverage bins*, and output *coverage bins* have been individually covered or not. The input coverage should be 100% because it depends on the static stimuli parameters and not on DUV's behavior. It is evident from the results in Table I, Table II, and Table III. The output functional coverage is dependent on the input stimuli signals and DUV's behavior. Our goal is to achieve 100% output coverage. If all the coverage bins are covered

```
coverpoint<double> A_cvp = coverpoint<double> (this,
bin<double>("low corner", 0.0),
bin<double>("0.5", 0.5),
....
bin<double>("4.5", 4.5),
bin<double>("high corner", 5.0)
);

coverpoint<double> F_cvp = coverpoint<double> (this,
bin<double>("low corner", 5e3),
bin<double>("8e3", 8e3),
....
bin<double>("17e3", 17e3),
bin<double>("high corner", 20e3)
);

coverpoint<double> input_power_cvp =
     coverpoint<double> (this,
bin<double>("20", interval(20:24.99)), // 4.99 dBm
bin<double>("25", interval(25:29.99)),
bin<double>("30", interval(30:31.99)), // 1.99 dBm
bin<double>("32", interval(32:33.99)),
bin<double>("34", interval(34:35.99)),
bin<double>("36", interval(36:37.99)),
bin<double>("38", interval(38:38.99)), // 0.99 dBm
bin<double>("39", interval(39:39.99)),
bin<double>("40", interval(40:40.99))
);
```

Fig. 5. LNA: input coverage coverpoints definition

TABLE IV
LNA GAIN (G) OUTPUT COVERAGE REPORT

	gain_cvp "gain_values"			
	Description	**Value**	**#Hits**	**Hit**
37.5%	low corner	[16.5: 16.7]	0	x
	17.0 dB	[17.0: 17.2]	0	x
	17.5 dB	[17.5: 17.7]	0	x
	18.0 dB	[18.0: 18.2]	21	✓
	18.5 dB	[18.5: 18.7]	0	x
	19.0 dB	[19.0: 19.2]	27	✓
	19.5 dB	[19.5: 19.7]	21	✓
	high corner	[20.0: 20.2]	0	x

resolution = 0.5 dB, *interval resolution* = 0.2 dB

in the coverage report, i.e., 100% coverage, nothing else is required to be done. Please note, 100% coverage only indicates that all the defined objectives (*coverage bins*) of the DUV characterization have been achieved. It does not necessarily mean that all DUV specifications have been verified. In case the DUV specification is not covered, there are three cases:

C1: The *resolution* of the static stimuli parameters is coarse (low), hence, the input stimuli signals are far apart. As a consequence, many output *coverage bins* are not hit, and coverage holes appear. In this case, the *resolution* should be increased, i.e., step size should be decreased.

C2: The *resolution* of the *output coverage collectors* is coarse (low) or the *bins* are not defined. Hence, it is not possible to capture DUV's output specifications. In this case, either new *bins* need to be defined (as per specifications), or the *resolution* needs to be refined, i.e., decrease step size.

C3: The DUV has a bug, and refining *resolution* or *interval resolution* of static stimuli parameters, or *output coverage collectors* has no effect. In this case, analyze the DUV implementation for possible bugs.

Case C2 has a higher priority than case C1. The reason is that if *bins* for a specification are not defined in the *output coverage collector*, the output behavior cannot be observed.

We observe that the functional coverage for gain (G) in our running example is not 100%, rather only 37.5% as shown in Table IV. According to Section IV-F case C1, to increase the output coverage, we need to refine the stimuli parameter resolution. We also observe that with the defined *resolution*, and *interval resolution* of gain, the bin for DUV gain, i.e., 19.9 dB, and the 1 dB compression point bin, i.e., 18.9 dB, do not appear as shown in Table IV. Please note, even if gain of 19.9 dB and 1 dB compression point were achieved in functionality, we have no way to capture it. Therefore, we have coverage holes. According to case C2, new bins need to be defined. For C2, we add new bins ranging from 18.8 dB to 19.9 dB with a resolution of 0.2 dB, and *interval resolution* of 0.1 dB. But we observe that the bins are still uncovered with the functional coverage of 41.66% (Table V). As per case C1, we increase

TABLE V
LNA GAIN (G) COVERAGE REPORT. CASE C2: ADDITION OF BINS IN GAIN COVERPOINT BETWEEN 18.8 *d*B AND 19.9 *d*B

	gain_cvp "gain_values"			
	Description	**Value**	**#Hits**	**Hit**
41.66%	low corner	[16.5: 16.7]	0	x
	17.0 dB	[17.0: 17.2]	0	x
	17.5 dB	[17.5: 17.7]	0	x
	18.0 dB	[18.0: 18.2]	21	✓
	18.5 dB	[18.5: 18.7]	0	x
	18.8 dB	[18.8: 18.9]	0	x
	19.0 dB	[19.0: 19.1]	9	✓
	19.2 dB	[19.2: 19.3]	0	x
	19.4 dB	[19.4: 19.5]	9	✓
	19.6 dB	[19.6: 19.7]	9	✓
	19.8 dB	[19.8: 19.9]	9	✓
	20.0 dB	[20.0: 20.2]	0	x

resolution gain= 0.2 dB, *interval resolution* gain= 0.1 dB

TABLE VI
LNA GAIN (G) COVERAGE REPORT. CASE C1: STATIC PARAMETER REFINEMENT ON INPUT STIMULI

	gain_cvp "gain_values"			
	Description	**Value**	**#Hits**	**Hit**
100%	low corner	[16.5: 16.7]	9	✓
	17.0 dB	[17.0: 17.2]	18	✓
	17.5 dB	[17.5: 17.7]	96	✓
	18.0 dB	[18.0: 18.2]	219	✓
	18.5 dB	[18.5: 18.7]	24	✓
	18.8 dB	[18.8: 18.9]	66	✓
	19.0 dB	[19.0: 19.1]	144	✓
	19.2 dB	[19.2: 19.3]	9	✓
	19.4 dB	[19.4: 19.5]	132	✓
	19.6 dB	[19.6: 19.7]	111	✓
	19.8 dB	[19.8: 19.9]	162	✓
	20.0 dB	[20.0: 20.2]	18	✓

resolution A= 0.1 V, *resolution* F = 1 KHz

TABLE VII
CROSS COVERAGE BUG-FREE DUV (EXCERPT) - CHECKER VS INPUT POWER VS GAIN

	checker vs input power vs gain		
	Description	**#Hits**	**Hit**
92.4%	Pass x 30 dBm x 18.9 dB	66	✓
	Pass x 32 dBm x 18.8 dB	66	✓
	Pass x 36 dBm x 18.65 dB	26	✓
	Pass x 40 dBm x 18.52 dB	26	✓

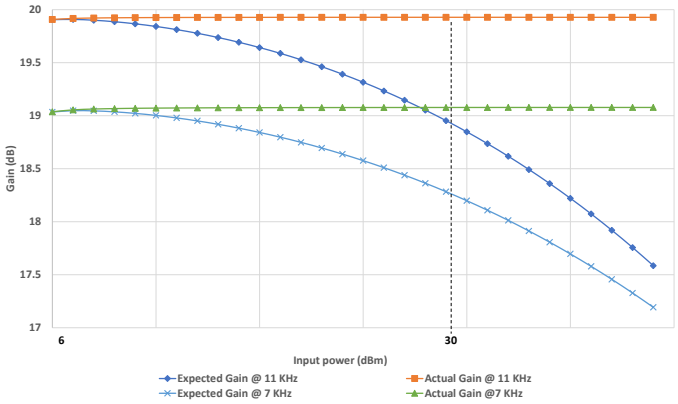

Fig. 6. Gain vs Input Power

the *resolution* (decrease step size) of *A* from 0.5 V to 0.2 V, and *F* from 3 KHz to 1 KHz. We observe that the output coverage increases to 46%. We further refine the resolution of stimuli parameter *A* to 0.1 V, but we don't observe any improvement in coverage. As per case C3, we analyze the DUV's implementation, and find a bug. Fig. 6 shows the gain of the buggy DUV (actual gain) and bug-free DUV (expected gain), where the LNA does not saturate with increasing input power. The gain is only shown for 2 input frequencies for ease of understanding. The expected gain is never observed, and hence, the checkers are also never triggered. Once the bug is fixed, the output coverage becomes 100% (Table VI), and the checker is triggered to verify the correct behavior. A part of cross-coverage report is displayed in Table VII which shows the cross-coverage between checkers, input power, and gain. The first row of Table VII can be interpreted as follows, the checker indicates the correct functionality of the DUV with *pass*, the input power at the time when the correct functionality was observed was 30 dBm, and the output gain was 18.9 dB. The functional coverage not only shows the exact behavior, but also the complete range of operation of the DUV, which is generally required.

V. Industrial Case Study

In this section we present a case study using an industrial RF transmitter and receiver model (Fig. 7) to show the efficacy of our proposed approach. The system uses LNAs at different positions in the systems. The AMS model is implemented in SystemC AMS, and the simulations are carried out using the commercial tool COSIDE [33].

The system models a complete RF transmitter and receiver structure. Transmitted symbols are transferred using a *Differential Quadrature Phase Shift Keying* (DQPSK) modulation. After an up-sampling of the encoded signal, the signal is mixed up to the transmission frequency. To model the complete transmission chain an *Additive White Gaussian Noise* (AWGN) channel model is used. Thereby, white Gaussian noise (white noise) is added to the transmitted signal. The receiver first amplifies the received signal using a LNA and then mixes it down from the transmission frequency during demodulation. Afterwards, a detector tries to decide which signals have been sent and maps them on the symbols.

For verifying the overall system a testbench was created, which sends random symbols over the RF transmitter, and compares the received symbols on the receiver side. Based on this, a bit error rate can be calculated. The verification goal is to evaluate the quality of the testbench, and to make sure all the functionality (w.r.t. specifications) of different models is covered, hence, functional coverage is used. The parameters of interest for our experiement are *gain, IIP3, 1 dB point*. The coverpoints are placed at the following points in the transmitter (Fig. 7) at the input and output of: LNA (*i_gain_cplx1*) with gain = $6dB$, IIP3 = $30dBm$, 1 dB point = $180dBm$, and input/output resistance = $100ohm$. In the receiver chain: 1) LNA (*i_lna_base_pb1*) with gain = $20dB$, IIP3 = $20dBm$, 1 dB point = $60.4dBm$, input/output resistance = $50ohm$, 2) LNA (*i_gain_cplx2*) with gain = $20dB$, IIP3 = $10dBm$, 1 dB point = $60.4dBm$, input/output resistance = $100ohm$. Coverpoints were created to check the *gain (G)* (dB), *1 dB compression point* (dBm), and *third-order input intercept point (IIP3)* (dBm) characteristics. The refinement parameters; *resolution*, and *interval resolution* for gain are $2dB$, and $0.5dB$, respectively, in first iteration. *resolution*, and *interval resolution* for other characteristics are $10dBm$, and $5dBm$, respectively, in first iteration. Fig. 8 shows the development of coverage for each LNA over three iterations using the proposed approach. In Fig. 8, each LNA model's coverage is shown with different color set in each iteration.

It was to be expected, that not all coverpoints used in block level testbench are covered. However, our approach provides insight on the DUV operation; 1) out of scope of the intended specifications, 2) coverpoints have to be refined with a different *resolution*, and *interval resolution*, as they are not detailed enough to capture the use case within the complex system. In each iteration, the *resolution*, and *interval resolution* were refined to precisely check the covered characteristics. Interestingly, the gain coverpoints in the transmitter chain behaved as expected. There was no non-linear behavior, hence, the total coverage remained low. As shown in Fig. 8, some of the IIP3 bins were hit but they were hit in the lower power ranges. Hence, the coverage was always low. But in the receiver chain, non-linearity in the LNAs was observed because of AWGN, and the 1 dB compression point was hit. Some coverage bins for IIP3 point were also covered as shown in Fig. 8. Fig. 8 shows the increasing progression of different characteristics as the refinement parameters; *resolution*, and *interval resolution*, are refined over the iterations. In summary, the proposed AMS functional coverage-driven characterization approach for RF amplifiers systematically increased the coverage to verify the RF transmitter and receiver models.

VI. Conclusion

In this paper we proposed the first AMS functional coverage-driven characterization approach which elevated the main concepts of digital functional coverage to the SystemC AMS abstraction. The approach targeted the characterization

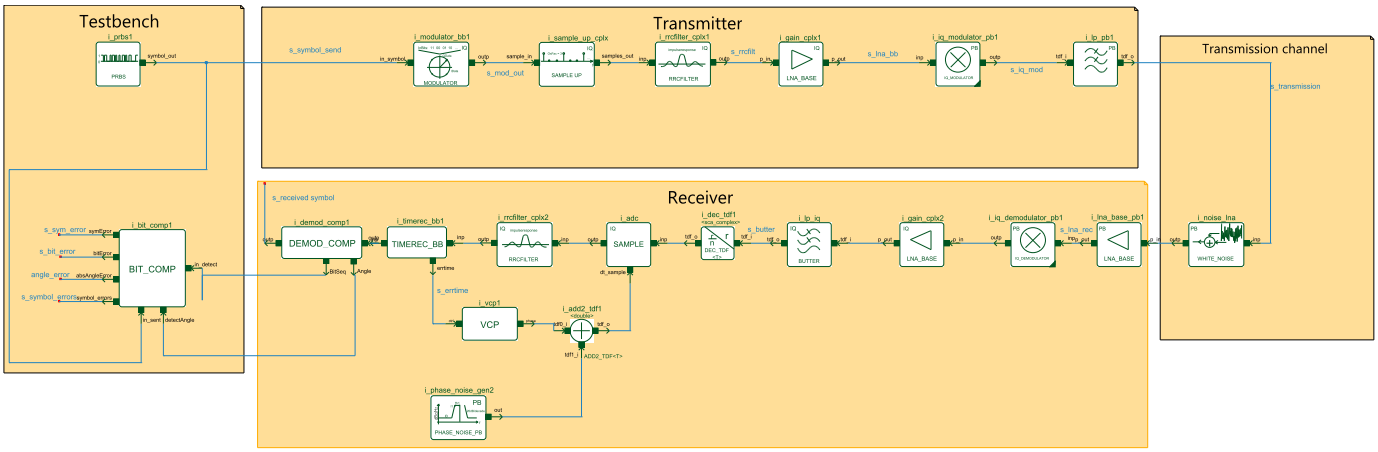

Fig. 7. Case Study: RF transmitter and receiver model

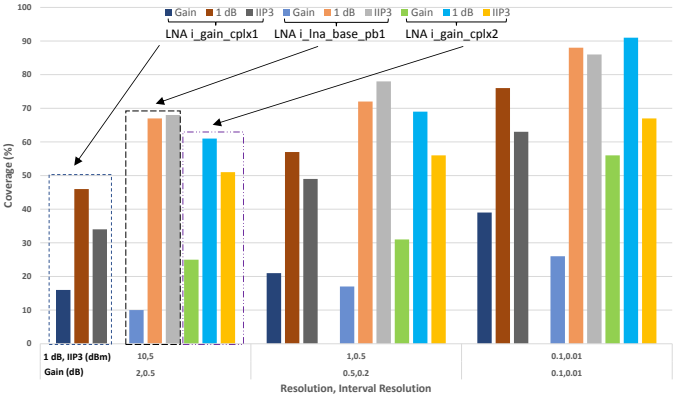

Fig. 8. Case Study - Coverage closure w.r.t. resolution

of the class of RF amplifiers in general, and we showed LNA as an example in particular. Two coverage refinement parameters; *resolution*, and *interval resolution*, were introduced to define the stimuli parameters and functional coverage bins at input and output. An AMS coverage analysis was proposed which crosses input and output functional coverage, and checkers, to systematically modify the refinement parameters to fully capture the specifications of the DUV. We showed the effectiveness and applicability of our approach on an industrial RF transmitter and receiver model. In future work, we plan to consider functional coverage-driven system-level verification of mixed-signal platforms, i.e. including software running for instance on top of RISC-V virtual prototypes such as [34], [35]. Furthermore, we plan to consider constrained random techniques to evaluate their effectiveness on the functional coverage [36].

References

[1] L. W. Nagel, "SPICE-simulation program with integrated circuit emphasis," *Electronics Research Laboratory, Univ. of California, Berkeley*, 1973.

[2] ——, "SPICE2: A computer program to simulate semiconductor circuits," *Ph. D. dissertation, University of California at Berkeley*, 1975.

[3] K. Karnane, G. Curtis, and R. Goering, "Solutions for mixed-signal SoC verification," *Cadence Design Systems*, 2009.

[4] C. Grimm, M. Barnasconi, A. Vachoux, and K. Einwich, "An introduction to modeling embedded analog/mixed-signal systems using SystemC AMS extensions," in *DAC*, vol. 23, 2008.

[5] M. Barnasconi, C. Grimm, M. Damm, K. Einwich, M. Louërat, T. Maehne, F. Pecheux, and A. Vachoux, "SystemC AMS extensions user's guide," *Accellera Systems Initiative*, 2010.

[6] M. Barnasconi, K. Einwich, C. Grimm, T. Maehne, and A. Vachoux, "Advancing the SystemC analog/mixed-signal (AMS) extensions," *Open SystemC Initiative*, 2011.

[7] K. Einwich, "Introduction to the SystemC AMS extension standard," in *DDECS*, 2011, pp. 6–8.

[8] Y. Chen, S. Vinco, E. Macii, and M. Poncino, "Fast thermal simulation using SystemC-AMS," in *GLSVLSI*, 2016, pp. 427–432.

[9] M. Lora, S. Vinco, E. Fraccaroli, D. Quaglia, and F. Fummi, "Analog models manipulation for effective integration in smart system virtual platforms," *TCAD*, vol. 37, no. 2, pp. 378–391, 2018.

[10] Y. Chen, S. Vinco, E. Macii, and M. Poncino, "SystemC-AMS thermal modeling for the co-simulation of functional and extra-functional properties," *TODAES*, vol. 24, no. 1, p. 4, 2018.

[11] F. Pêcheux, C. Grimm, T. Maehne, M. Barnasconi, and K. Einwich, "SystemC AMS based frameworks for virtual prototyping of heterogeneous systems," in *ISCAS*, 2018, pp. 1–4.

[12] M. Hassan, D. Große, H. M. Le, T. Vörtler, K. Einwich, and R. Drechsler, "Testbench qualification for SystemC-AMS timed data flow models," in *DATE*, 2018, pp. 857–860.

[13] Y. Chen, D. Baek, J. Kim, S. Di Cataldo, N. Chang, E. Macii, S. Vinco, and M. Poncino, "A SystemC-AMS framework for the design and simulation of energy management in electric vehicles," *IEEE Access*, vol. 7, pp. 25 779–25 791, 2019.

[14] *IEEE SystemVerilog*, IEEE Std. 1800, 2005.

[15] C. Spear, *SystemVerilog for verification: a guide to learning the testbench language features*. Springer Science & Business Media, 2008.

[16] M. Barnasconi, "SystemC AMS extensions: Solving the need for speed," *DAC*, 2010.

[17] A. Piziali, *Functional verification coverage measurement and analysis*. Springer Science & Business Media, 2007.

[18] S. Tasiran and K. Keutzer, "Coverage metrics for functional validation of hardware designs," *IEEE Design and Test of Computers*, vol. 18, no. 4, pp. 36–45, 2001.

[19] D. Große, U. Kühne, and R. Drechsler, "Analyzing functional coverage in bounded model checking," *TCAD*, vol. 27, no. 7, pp. 1305–1314, Jul. 2008.

[20] Mentor Graphics, "Coverage cookbook," 2013, available at https://verificationacademy.com/cookbook/coverage.

[21] W. Hartong, N. Luetke-Steinhorst, and R. Schweiger, "Coverage driven verification for mixed signal systems," in *ANALOG Conference*, 2008.

[22] A. Fürtig, G. Gläser, C. Grimm, L. Hedrich, S. Heinen, H.-S. L. Lee, G. Nitsche, M. Olbrich, C. Radojicic, and F. Speicher, "Novel metrics for Analog Mixed-Signal coverage," in *DDECS*, 2017, pp. 97–102.

[23] D. Haerle, "Trends and challenges in Analog and Mixed-Signal verification," in *FAC Keynote Address*, 2018.

[24] H. Azatchi, L. Fournier, E. Marcus, S. Ur, A. Ziv, and K. Zohar, "Advanced analysis techniques for cross-product coverage," *TC*, vol. 55, no. 11, pp. 1367–1379, 2006.

[25] F. Haedicke, D. Große, and R. Drechsler, "A guiding coverage metric for formal verification," in *DATE*, 2012, pp. 617–622.

[26] M. Hassan, V. Herdt, H. M. Le, M. Chen, D. Große, and R. Drechsler, "Data flow testing for virtual prototypes," in *DATE*, 2017, pp. 380–385.

[27] J. Parky, S. Madhavapeddiz, A. Paglieri, C. Barrz, and J. A. Abrahamy, "Defect-based analog fault coverage analysis using mixed-mode fault simulation," in *IMS3TW*, 2009, pp. 1–6.

[28] A. Furtig, S. Steinhorst, and L. Hedrich, "Feature based state space coverage of analog circuits," in *FDL*, 2016, pp. 1–7.

[29] M. Hassan, D. Große, H. M. Le, and R. Drechsler, "Data flow testing for SystemC-AMS timed data flow models," in *DATE*, 2019, pp. 366–371.

[30] A. Ain, A. A. B. da Costa, and P. Dasgupta, "Feature indented assertions for analog and mixed-signal validation," *TCAD*, vol. 35, no. 11, pp. 1928–1941, 2016.

[31] A. Ain, A. Mambakam, and P. Dasgupta, "Feature based coverage analysis of AMS circuits," in *ISVLSI*, 2018, pp. 423–428.

[32] K. Kundert, "Accurate and rapid measurement of IP2 and IP3," *The Designers Guide Community, www. designersguide. org/Analysis/intercept-point. pdf*, 2002.

[33] "Coside®," http://www.coseda-tech.com.

[34] V. Herdt, D. Große, H. M. Le, and R. Drechsler, "Extensible and configurable RISC-V based virtual prototype," in *FDL*, 2018, pp. 5–16.

[35] http://www.systemc-verification.org/riscv-vp.

[36] T. Vörtler, K. Einwich, M. Hassan, and D. Große, "Using constraints for SystemC AMS design and verification," in *DVCon Europe*, 2018.