Ph.D. Dissertation

Parameter Coverage Analysis on Simulation of Analog Functional Models

아날로그 동작 모델의 모의 실험에 대한

파라미터 커버리지 분석

August 2019

Department of Electrical Engineering and Computer Science
Seoul National University

Jiho Lee

Parameter Coverage Analysis on Simulation of Analog Functional Models

지도 교수 김 재 하

이 논문을 공학박사 학위논문으로 제출함
2019 년 8 월

서울대학교 대학원
전기·컴퓨터공학부
이 지 호

이지호의 박사 학위논문을 인준함
2019 년 8 월

위 원 장 정 덕 균 (인)

부위원장 김 재 하 (인)

위 원 최 기 영 (인)

위 원 김 태 환 (인)

위 원 박 명 재 (인)

Abstract

This dissertation proposes a way to quantify the coverage of functional simulations with analog models. Specifically, the presented analysis can indicate whether a given simulation scenario is adequate to verify the correctness of the parameter values used in the analog functional models such as gain, offset, bandwidth, etc. The proposed coverage analysis derives the sensitivity of the simulated responses with respect to each parameter variation and determines the coverage by checking whether the sensitivity is greater than a threshold.

The main contribution of this paper is twofold. First, we derive the sensitivity measure function which can measure the relevancy of an input signal to a block which can have various scale and shape. Second, we determine the sensitivity thresholds based on the probabilistic input signal space and present information-maximizing principle based on the information theory. To apply the proposed analysis for various instances of analog model blocks, we standardize primitives blocks for analog and mixed-signal circuits and derive the actual sensitivity functions and decision threshold for each primitive. The primitives include scaler, adder, slicer, filter and D/A, A/D converters.

The proposed metric is evaluated on the typical simulation scenarios of wireline/wireless communication blocks, such as equalizers, high-order filters, and feedback modulators. The result reveals that actual coverage on each parameter provided by each of the simulation stimulus. Also, the experimental studies using the behavioral model of an RF analog front-end show that the

proposed coverage metric can vary from 40% to 90% depending on the simulation scenarios, and correlates well with the simulation's ability to reveal parametric design bugs, with the correlation coefficients ranging 0.60 ~ 0.83.

Keywords : Simulation-based verification, Analog functional model, Parameter coverage analysis, Template-based modeling, Waveform sensitivity analysis, Probabilistic input space

Student Number : 2013-20860

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The design implementation of analog blocks is complex in recent mixed-signal ICs, resulting in a high possibility of possessing design errors. It demands a thorough verification process for analog blocks, as mis-identified errors in analog blocks can break down the function of entire IC, despite the digital blocks are correctly designed and verified with the mature verification flow.

In practice, most verification approaches are to run simulations using functional models of analog and digital blocks so that the analog blocks are verified for system-level operation. In such simulation-based verification, the lack of simulation coverage analysis on analog blocks are reported as a major source of bug-escapes during verification [1], [2].

This chapter will discuss more on the source of verification complexity of analog blocks, the typical design errors and the importance of coverage analysis in the current verification approaches. Then, we briefly discuss the main contribution and organization of this dissertation.

1.1 Complexity of Analog Blocks in Mixed-Signal ICs

Nowadays, the market of mixed-signal integrated circuits (ICs) exhibits various requirements arising from large number of different customer applications. To increase their market share, the suppliers of mixed-signal ICs release a single-chip product that satisfies several standards. For example, a wireless transceiver IC in mobile devices is designed to support all the 2G - 5G standards in mobile communication [3] or several Wi-Fi standards and the Bluetooth protocols [4]. Similar examples can be found in configurable IP blocks for wired communications supporting PCIe, SATA, USB protocols [5] or sensor interface ICs reading different types of physiological signals [6]. As a result of pursuing this multi-modality, recent mixed-signal ICs contain multiple analog blocks in parallel and configure them to support each of different requirements.

Along with the need of multi-modality, the continuously increasing requirements on the low power consumption and the high bandwidth also poses design complexity in analog blocks. The operation parameters of analog blocks (e.g., gain, bandwidth) are extensively calibrated to compensate the negative effect of process, voltage and temperature variations during fabrication, or adaptively controlled for the operating condition of system. In industrial ICs, this controllability is implemented by using variable resistors, capacitors, and voltage and current references. Dozens of digital control signals and model parameters exist even for a single analog amplifier block.

This design complexity in analog blocks made the implementation of analog designs more error-prone, but the design errors are often missed due to insufficient verification process. One of the mostly reported errors are functional errors due to simply swapped bus signal or inverted logic values [7], arising on digital control signals for the analog blocks. Unlike to marginal shortage on the performance specification for analog, such as noise characteristics or linearity, the functional errors can be caused by simple mistakes.

In the hierarchical flow of design and verification of analog blocks, those errors are not revealed or occur in the stage of designing individual blocks. Instead, the errors occur and are revealed in the stage of merging multiple analog blocks and digital blocks in system-level. The result of these errors is non-recurring cost spent for previous tape-out and market delay, which demands thorough verification especially for system-level design implementation.

1.2 Simulation-Based Verification and Coverage Analysis

Design verification is to establish certain property of a design implementation under certain assumption. The verification is practically performed with running simulations, which is to apply input stimuli to the design under verification (DUV) and calculate the output responses. The property to be established by verification is given as the specification on the output signals. Also, the assumption on the design is determined by the representation of the actual circuit, which are the models of circuits used for simulation.

Different simulation models provide various levels of accuracy at the expense of simulation time. Among them, the SPICE circuit models are referred to as the golden level of design abstractions for analog circuits. However, system-level verification requires simpler models to reduce the simulation execution time yet preserving the functional feature of analog circuits.

The functional models for analog blocks represent the analog functions in simplified way, such as ideal multiplication, summation, filtering of analog signals. These analog functional models can be simulated together with digital models to validate the digital controllability on analog performances or the intertwined feedback operation between analog and digital blocks. Up to now, various attempts were made to describe analog functional models in the digital hardware description language, such as the SystemVerilog, and simulate them with the digital blocks in conventional digital simulators.

Most previous efforts for enhancing the quality of simulation-based verification were to enhance the speed of simulation, such as developing a fast yet accurate simulation algorithms, such as the fast SPICE simulators [43], or deriving a novel representation of signals applicable for a specific design [44]. Unfortunately, the current cases of verification failures for analog and mixed-signal ICs are hardly expected to be overcome by simply enhancing the speed of simulation [45]. Instead, the unsystematic flow of verification is reported as a major source of bug-escapes during analog verification [1], [2], [7].

As an effort for strengthening analog verification flow, this dissertation addresses coverage analysis on analog blocks. In simulation-based verification, the main purpose of coverage analysis is to answer the question; "How thoroughly the design was simulated". In answering it, the coverage analysis assesses the adequacy of a test-bench, or the set of input stimuli applied to the DUV with a systematic evaluation method.

Figure 1.1 exemplifies a typical simulation testbench for an analog functional model, which highlights the role of coverage analysis. In a typical testbench, there are several components that comprise a complete verification process; Stimuli generator provides input signals for digital and analog blocks. Simulator computes the response of models. Response monitor compares the simulation results against an expected result. Finally, coverage analyzer measures how much of the design is exercised during simulation.

Coverage analysis consists a part of iterative process of generating input stimuli, running simulation, check output validness. If the coverage is low, there are chances of a design error to be not exposed and the stimuli generator

provides novel inputs to the model. On the other hand, if coverage is high and all of the simulation results are validated as correct by the response monitor, then the verification is completed.

Unlike the verification approaches depending on the human-written test cases, the coverage analysis provides an objective and quantitative evaluation on the simulation. Especially, the coverage analysis can identify the part of design accidentally missed by a thorough verification engineer. But in current state of the art, coverage analysis on analog blocks is not widely proliferated in contrast to the well-established digital verification flow that are equipped with several coverage metrics [9].

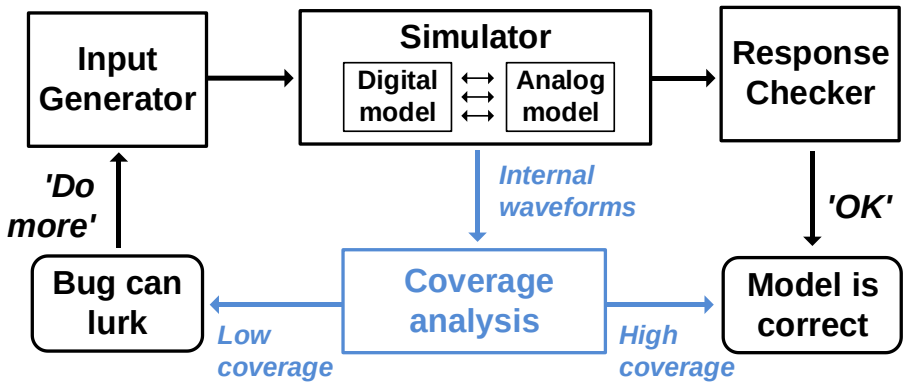


Figure 1.1. Role of the coverage analysis in a typical simulation testbench, which conceptually consists of input generator, simulator, response checker, and the coverage analyzer.

1.3 Thesis Contribution and Organization

This dissertation presents a parameter coverage analysis for analog functional models. Specifically, the presented coverage analysis aims to decide the coverage of each functional model parameter in analog blocks with yes/no result and in an automatic way.

The main contribution of this dissertation is on deriving sensitivity functions and proposing a basis for finding decision threshold, which enable automatic structural coverage analysis on analog functional models. The proposed sensitivity functions can evaluate the functional relevancy of input signal for various analog blocks, despite that analog models exhibit various types of functionalities and analog signals are continuous. Also, the decision thresholds are derived in closed-form expression of the parameter to cover, based on a general space of input signals for each of analog functional blocks.

In remaining of this dissertation, we first review on the motivations of existing coverage metrics and propose the parameter coverage metric (Chapter 2). Then, we describe the template-based expression of analog functional blocks (Chapter 3). For each of the templates, we derive sensitivity measurement functions and decision threshold (Chapter 4 and 5). Afterwards, the experimental result of applying proposed coverage analysis is presented with block-level and system-level analog functional model examples (Chapter 6), along with the future works. Then we conclude the paper (Chapter 7).

Chapter 2

Parameter Coverage Metric for Analog Functional Models

A coverage metric is simply defined as the ratio of two numbers. The denominator is the number of total goals, that are desired to be achieved from simulation. The numerator is the number of achieved goals actually from a given simulation. Thus, the quality or usefulness of a coverage metric is determined from the definition of coverage goals. In fact, digital verification has established various coverage metrics and uses under different verification interests [9]. For the existing analog coverage metrics [11] - [23], the definition of coverage goals is also the starting point of the further analyses.

In following subchapters, we first review the existing coverage metrics in digital and analog verification, specifically with the different motivations coming from design-specific verification interests. Then, we propose the coverage metrics for analog functional models.

2.1 Analog Functional Models

An analog functional model is an abstract representation of a transistor-level analog circuit implementation, that simplifies the relationship of input and output signal of the circuit. In hierarchical design flow of analog blocks, the functional models have two different roles. First, the models define the design goals for analog circuits before implementation, where the interest is to find design goals for each block so that the system-level performance specification would be achieved. The second role of function models is characterizing a circuit to construct a system model in bottom-up way. Then, the system-level design implementation containing both analog and digital blocks is verified.

Analog functional models have model parameters, such as the gain of an amplifier, the bandwidth of a filter, the conversion weight for a digital-to-analog converter, and the conversion threshold for analog-to digital converters. The model parameters are important since they reflect the design intent of analog circuits. Specifically, in recent designs, the parameters are made controllable by digital control signals. As a result, large number of possible gain or bandwidth values exist for an amplifier, filter, or equivalently, large number of conversion weight parameters used for mapping control bits to the actual value of that gain and bandwidth.

The parameter values of analog functional models are error-prone, since analog design flow is mostly based on running simulations and extracting parameters from simulation signals manually. In contrast to digital design flow, which proves equivalence of different design representations (e.g., behavioral

level, register-transfer level, and gate level design implementations), analog designs are validated by comparing waveforms or signal features with a tolerance value. As a result of an insufficient verification, the equivalence between analog models and circuits can be broken or the system-level specification on analog functional models can be violated.

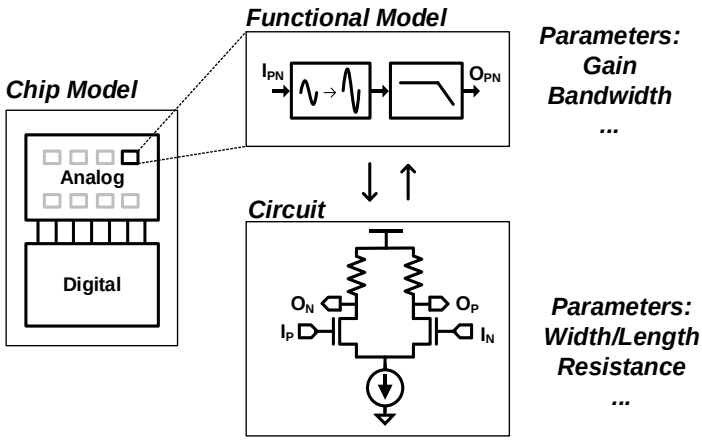


Figure 2.1. Analog functional models simplify the transistor level implementation of a circuit, used for constructing the system model of a mixed-signal IC.

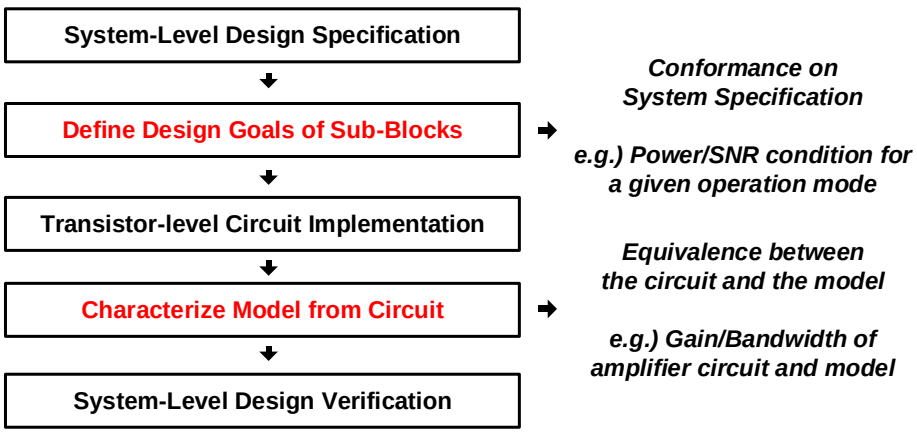


Figure 2.2. Design verification problems related to analog functional models in hierarchical design flow.

2.2 Challenge in Finding Design Errors of Analog/Mixed-Signal Functional Models

To find an error in DUV with simulations, the output signal of a block that containing the error should be clearly different from original signal to reveal the presence of an error. Also, as selective signals in the simulation results on entire DUV are monitored and checked, the effect of the error should propagate to that representative signals under monitoring.

In purely analog systems, any change inside the system always change the output response of the system, in contrast to digital systems where an internal error can be completely blocked by Boolean logic operations. However, the amount of error in analog continuously changes with respect to the change of both the input and parameter. This continuity poses challenge together with the practice of checking analog simulation results, which is to compare an output signal against a reference waveform, or to compare a feature value of signal to a refence one, with some tolerance values. To detect a design error by simulation, the effect of the error should *clearly* change the signals that are checked after simulation. But deciding the level of error becomes ambiguous.

Figure 2.3 illustrates two cases, where the effect of error can fade after passing an analog adder or analog filter. Figure 2.3(a) illustrates that the Block 1 has a design error, resulting that the upper input of the adder block driven by an undesired signal (red). However, the adder receives a stronger signal through the other input, reducing the relative change in the output of the adder. Then, further propagating through Block 3, the erroneous behavior of Block 1 is may

be not detected. For example, the range of probed signal can be checked against the range of golden answer with a tolerance, which falsely ignores the effect of error. In this case, the output of Block 2 should be weak to increase the possibility of detecting the error at the checker.

Figure 2.3(b) illustrates another case, where an error occurred in Block 1, which is a digital block that maps digital inputs to the bandwidth value of the filter. If the analog input of the filter is a sinusoidal signal with a far lower frequency than the bandwidth, then the output of the filter would change little. Then, after propagating this little effect of error through Block 3, the spectrum checking on the final output of is not probable to detect the error.

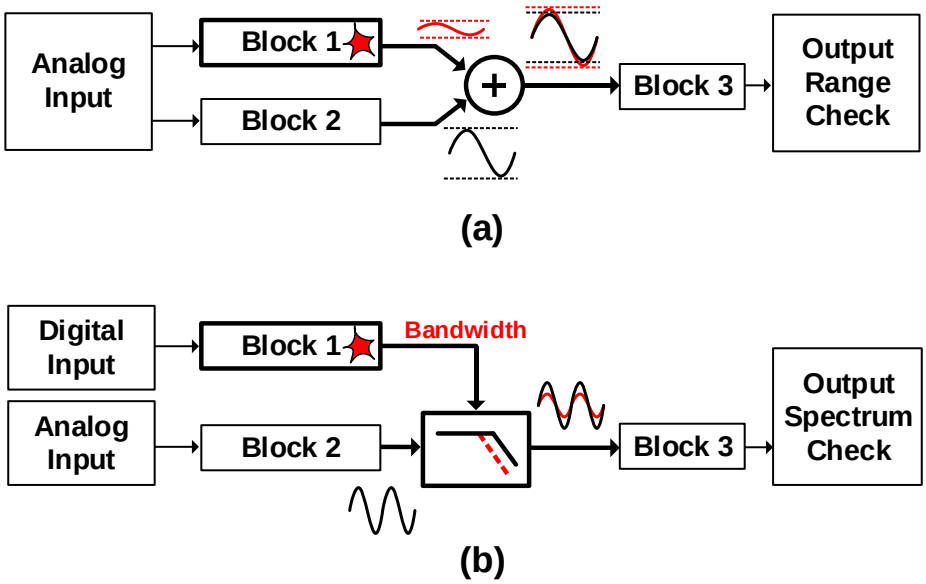


Figure 2.3. Design verification problems related to analog functional models in hierarchical design flow.

For both cases, the propagation of error depends on all the selection of external analog or digital inputs and the operation of intermediate blocks (Block

1-3). As a result, it becomes difficult to find a sufficient set of simulation inputs that makes all the error in a design to be revealed at the local output of the block and be propagated to the monitoring point of output.

Along with the phenomena in analog functional blocks, being the 'mixed-signal' poses other challenges. The design errors can occur in digital blocks or at the interface of digital and analog blocks, but the effect is observed through the operation of analog blocks. Also, mixed-signal ICs use analog multiplexers or switches, for block enable/disable functions or for inserting test pins. This existence of switches becomes another source of losing the effect of error through propagation, exhibiting *digital* phenomena of completely blocking the propagation of the effect of errors.

Specifically, the misdetection of catastrophic design errors is highly likely due to these switching effects. While these block on/off functions can be simply modeled with output enable switches, but the switches are distributed among cascaded analog blocks, forming large number of paths constructed by cascaded analog blocks under verification.

Interestingly, typical errors occurring at the interface between digital and analog blocks are repeatedly pointed out in literature over time [42], [7]. Being apart from almost 20 years, the two papers state a same problem that the inversion of MSB/LSB of control codes for analog blocks is a major source of failing mixed-signal ICs. As those types of errors are catastrophic, the output of the block near the point of the error will be clearly different from the desired value. It implies that the failure of finding those errors comes from the loss of the effect of error through propagation.

2.3 General Motivation of Simulation Coverage Analysis

An ideal level of sufficiency can be achieved in simulation-based verification, by applying all possible input stimuli that a DUV may take. Figure 2.4 illustrates this. If all the output responses corresponding to the all input stimuli are correct, where the correctness is given from the specification on the design, then the correctness DUV is established. For example, the functionality of a digital block of N input is fully verified by applying all 2^N input codes and monitoring the output.

Another approach is to list all possible design errors and observe whether a simulation output is changed when an error is occurred in design. Then, a sufficient set of simulation inputs is defined as the set of inputs that can collectively detect all the possible design errors. As this set is a part of the abovementioned all possible inputs, it indirectly verifies the design from that if any error is in the design, then it would be detected by some input in the set.

However, applying all possible inputs or error cases can result in too much time for simulation. As recent analog functional blocks also include the digital blocks controlling the analog parameters, the number of possible combinations of digital inputs is exponential. Also, for each single case of digital input, the continuous analog input signals comprise infinite number of cases.

Therefore, in practice, coverage metrics are used as a workaround which define a subset of all possible cases and counting the number of cases that are exercised by simulation. For example, one can analyze the functional coverage

of the abovementioned digital design by selecting a subset of all 2^N input cases and counts the number of inputs that are applied to the DUV during simulation. Likewise, fault coverage metrics defines a set of possible design faults and counts the number of faults detected by simulation. Figure 2.5 illustrates the approach of various coverage metrics.

Interestingly, structural coverage metrics are not derived from the all possible cases or inputs or fault models. Instead, the design is understood as a structure of sub-elements and the coverage is locally decided on each part of design. This implies white-box analysis for the design, unlike the functional coverage metrics that regard the design as block-box and observe the signals on the boundary of design. Specifically, structural coverage metrics are not necessarily aware of correctness of output signal and always separate the role of checkers from the coverage analysis. The motivation of structural coverage is to find completely un-simulated part in a DUV, in cooperation with the checker that validates the correctness of the output signal for an input.

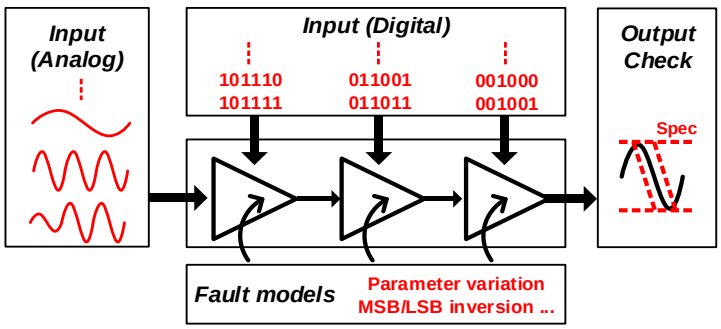


Figure 2.4. Simulation for analog functional models can have ideal sufficiency with applying all possible digital/analog inputs or all possible design errors.

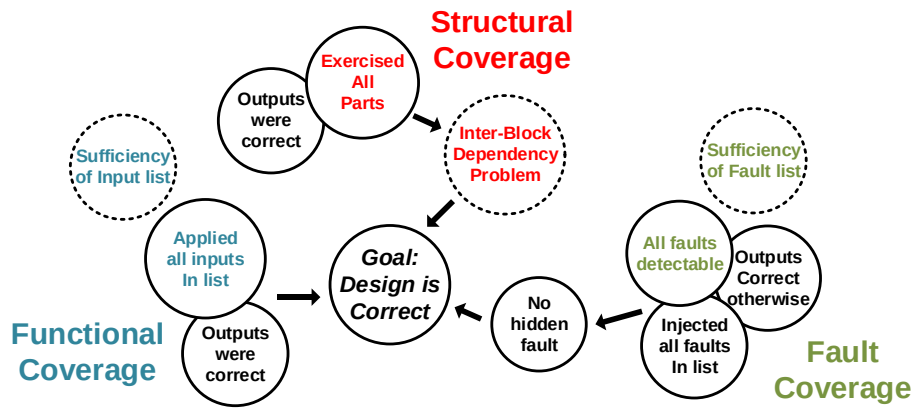


Figure 2.5. Difference of coverage metrics in establishing that a design is correct

2.3.1 Motivation of Digital Coverage Metrics

One motivation of digital coverage metric came from the discrete nature of digital designs. For example, a digital design which has 10-bit input takes 1024 total cases of possible input codes. In verifying this design, observing the success in 1023 cases does not guarantee the success on the remaining 1 case. The design implementation using look-up tables mostly highlights this property. In such table, each output value can be assigned without any correlation and corner-case of failure can exist. One extreme example is the notorious recall of Intel Pentium processor in 1994, known to be caused by an error in look-up table of programmable logic array [10]. The coverage analysis on those designs is toward checking every possible cases of the combination of Boolean signals.

This discrete nature of digital design is a common motivation for digital coverage metrics [9], such as code coverage, FSM coverage, functional coverage. The code coverage metrics measure the portion of executed statements or expression in an HDL source code, as the correctness of a line

can be independent to the others and the un-executed parts of code can contain undetected errors. Likewise, the FSM coverage measure the portion of visited states of a design, as an error can uniquely exist in any of discrete states of a digital design. In the sense of the previous categorization, FSM coverage and various code coverage metrics fall into structural coverage metrics. The functional coverage also ultimately tries to cover every individual cases of functional specification on the input and output of a design, as the functional correctness for each input cases may not be correlated. The motivation of digital fault coverage came from another perspective. A fault (e.g., stuck-at-1/0 or stuck-at-open/short) in a digital design can be concealed during the operation of logic gates (e.g., NAND, NOR), being exacerbated by the depth and complexity of pipelined logic stages.

In importing the digital coverage metrics to analog, these motivations should be carefully reviewed. As analog signals and functions are continuous, observing the correctness of a single case of input can be generalized for the case of applying other input. Another difference is on the topology shown in the connection of analog blocks. The total number of unit function blocks in the signal path from system-level analog input to analog output is usually far less than the number of digital gates that forms a logic path. Also, the fan-in and fan-out structure of the signal connections of analog blocks are more regularized than that in digital designs. Due to these differences, a straightforward extension of digital coverage metrics might not proper in defining analog coverage metrics.

2.3.2 Existing Coverage Metrics for Analog

The FSM coverage in digital verification can be extended to the state-space coverage on analog circuits [21], [14], [15], [16] that have state variables which determines the operation of circuit. These metrics have found usefulness in verifying nonlinear circuits, which operates well in most of its states but occasionally breaks when entering specific region during operation. Example verification problems in literature are global convergence of ring oscillators [46] or checking the operation region of nonlinear circuits [47].

These coverage measurements on state space are useful when one does not know which part of the state space of the circuit will be problematic before running simulations. However, for linear system models, the state space coverage becomes less meaningful as the operation of the system is not critically changed due to the value of state variables.

Another approach of importing state space coverage in digital is to build a digital-equivalent model of an analog block [26]. The model can be represented in z-domain signal processing and then the coverage is measured on digital signals. But this approach may discretize an analog signal into unnecessarily fine intervals and regard each sample values of digitized analog signals as independent ones. Then, the temporal correlation existing in two sample values of an analog signal can be lost, generating an excessively large number of coverage goals.

Application papers from industries mostly use signal features to define and measure the coverage on analog models [18] - [23]. These approaches are similar to the functional coverage metrics in digital, where a verification

engineer describes the cases of features that should be observed from a signal waveform. For example, in simulating the step response of an analog filter with variable bandwidth controlled by digital signals, a verification engineer states the coverage goal as the set of values for the rise time of output.

The strength of this approach is that stating the coverage goals can be highly customized for the target DUV, with specifying important signal features in design stages such as range of signal, rise or fall time of signal, settling time, or frequency spectrum. However, this approach requires an expert verification engineer to investigate the target design for selecting a signal net to observe and defining the relevant feature to extract. Also, the resultant coverage analysis method may be not reusable on different design or under different simulation stimuli, due to the inability of extracting features.

The parameter coverage for analog functional models can be defined based on the fault coverage. In fault coverage analysis, a design error (or instance of a fault) is intentionally occurred in the DUV and the simulation is performed, where the computation of signals implies the computation of propagation of the effect of error in DUV. As this approach is to run actual simulations on DUV, it automatically considers the operation of all intermediate blocks from the source block with fault to the sink point where a signal is probed and checked. Despite it can be computationally intensive, injecting faults and computing their propagation is a solid solution for evaluating the ability of a simulation in revealing design errors.

However, proper fault models on analog parameters is not yet established. Especially, the parametric variation of analog blocks has continuous amount,

which results in a new problem of finding the tolerable range of parametric variation. That is, if a certain amount of parameter variation yields only tolerable output changes for all possible inputs, then that parameter variation is not a fault and vice versa. As a result, performing fault coverage is to find an input signal in given space of all possible input signals that maximize the change in output, and check whether the maximum indeed violates the tolerable region or not. Consequently, a same amount of parametric deviation can be marked as either a fault or not, depending on the set of possible input signals, output measurement item and the tolerance value on it.

The structural coverage is helpful in finding 'completely missed' part during simulation. However, the structural coverage for analog functional models are less available except for [11], [12] presenting the usage and extension of digital code-coverage metrics for analog models. The motivation of structural is in opposite direction to the fault-injecting simulation methods, where the simulator computes the propagation of error through the all intermediate blocks. The structural coverage analysis for analog functional model parameters demands a coverage decision framework independent to human-described lists of signal features or fault models.

2.4 Proposed Parameter Coverage Metric

We decide the parameter of an analog block only observing the input and output of the block. The operations of the neighbor blocks are not considered intentionally. The proposed coverage metric first measures the relevancy of a simulation input for a functional model by the amount of change in the output response of a block. Specifically, we express the relevancy as the sensitivity of output to a parameter. Then, the coverage decision is to compare the level of sensitivity to a threshold.

For a generic block with a parameter p, input $x(t)$ and output $y(t)$, the coverage decision rule is stated with a sensitivity measure function S and a decision threshold θ:

$$Cov(p, x(t)) = "S(p, x(t)) > \theta" \tag{2.1}$$

The result of coverage decision, Cov, is a Boolean variable taking True or False value representing that p is covered by simulation input $x(t)$ or not, respectively. Note that $y(t)$ disappeared in the expression since it is completely determined by the input and the parameter value.

We commonly apply this coverage decision rule for every block in a system model, which is enabled by standardizing the type of analog functional blocks and deriving sensitivity measure functions and threshold values. Figure 2.6 illustrates the overall flow of proposed parameter coverage analysis.

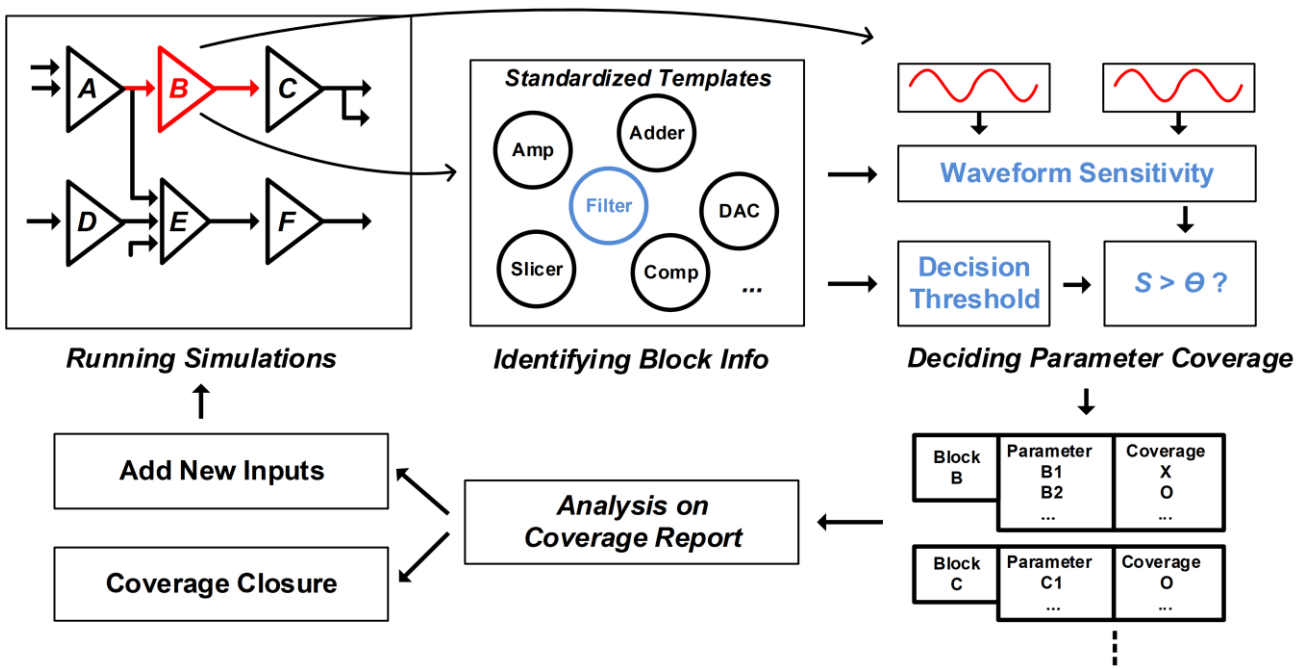


Figure 2.6. The overall flow of proposed parameter coverage analysis.

Chapter 3

Model Representation

Previous chapter derived the generic form of parameter coverage analysis for analog blocks. The actual coverage analysis depends on representation of analog models, especially that the proposed metric targets automatic decision on each model parameters. This chapter discusses on representing analog models using template blocks. As linear system abstraction is proved to express various analog circuits [24], [25], [37], [38], [39], the templates discussed in this chapter are mainly based on the linear system models.

3.1 Template-Based Modeling

Two strategies exist for describing an analog functional model; (1) instantiating blocks from a library and connecting them (2) describing signal computation in procedural way with mathematical expression. These two methods essentially represent same thing, yet the difference is that the former explicitly sets the boundary of a block. We take the former approach, as the block standardization and the expression of boundary provides advantages on our coverage analysis.

The former approach is referred to as template-based modeling, where a template has the functional description on the input/output mapping (e.g., summation, filtering), the parameters related to the function (e.g., weight, bandwidth). Also, the templates distinguish the direction of ports of block as input, output or in-out. By doing so, the connection of templates can easily represent the direction of signal propagation through models. For example, the a weighted summing operation, expressed as "$y = 2x_1 + 4x_2$", can be described with using a parameterized 'adder' template pre-defined in a library, or using procedural descriptions and basic tokens, such as '+' and '*', as in Figure 3.1.

For the first expression (Expr.1), we can easily apply block-wise coverage analysis for the instance "Inst1" that clearly declares signal directions parameter values. This easiness is also true for the second expression (Expr. 2), as we can interpret that the variable literals in right-hand-side (x1, x2) and the left-hand side (y) forms an adder expression, However, the third expression (Expr.3) needs more complicated interpretation on the source code. Expressing

other functions like frequency filters will be more complicated in using operator for derivative, integration and expressing the internal state variables.

In short, template-based modeling makes it easier to perform and implement block-based coverage analysis, at the expense of limiting the freedom in describing a model. Given model description and simulation waveforms, we can easily identify the template type of an actual block instance and read the input/output signal waveforms to perform the required computations.

In a simulator or description language, a template can be an any module or a block that contains and wrapping the built-in operators or signal processing routines for the simulation of analog functional models. Yet, the problem is on defining a compact library of templates that are sufficient to describe various analog functional models, with small number of templates that are exclusive to each other as much as possible.

```
Expr.1: Template-based description
Adder #(.weight('{2, 4})) Inst1(.out(y), .in({x1, x2});

Expr.2: Procedural description 1
assign y = 2*x1 + 4*x2;

Expr.3: Procedural description 2
assign y_1 = 2*x1;
assign y_2 = 4*x2;
assign y = y_1 + y_2;
```

Figure 3.1. Describing the functional model of an analog adder in block-instantiation or procedural description.

3.2 Analog Function Primitive

We state the templates for building analog functional models, which we call function primitives. Table 3.1 lists the functional expression of the input and output of each primitives with their parameters. In the following subsections, each of the primitives will be discussed in more detail.

Table 3.1. Description on the function primitives as common building blocks for analog functional model.

Type	Function description	Parameter(s)
Scaler	$y(t) = w \cdot x(t)$ $x, w \in \mathbb{R}$	w
Adder	$y(t) = \sum_{i=1}^{N} w_i x_i(t)$ $x_i, w_i \in \mathbb{R}$	$\{ w_i \}$'s
Digital-to-Analog Converter	$y(t) = \sum_{i=1}^{N} w_i x_i(t)$ $x_i \in \{0,1\}, w_i \in \mathbb{R}$	$\{ w_i \}$'s
Filter	$Y(s) = \frac{\prod_{j=1}^{N_z}(1 + s/z_j)}{\prod_{i=1}^{N_p}(1 + s/p_i)} X(s)$ $p_i, z_i \in \mathbb{C}$	$\{ p_i \}$'s, $\{ z_i \}$'s
Slicer	$y(t) = \mathrm{sgn}(x(t) - V_{TH})$	V_{TH}
Comparator	$y(\tau) = \mathrm{sgn}(x(t_n) - V_{TH})$ for $t_n < \tau < t_{n+1}$	V_{TH}

3.2.1 Adder and Scaler

An analog adder takes multiple inputs and generates the output by weight summation of the inputs. The model parameters of the adders are the summing weights, which are assumed to be arbitrary real values.

One example of using adders is for representing current-summing junctions in circuits. For example, the summing junctions of a decision feedback equalizer (DFE) or a feed-forward equalizers (FFE) in high speed links are represented by the adder primitive [48]. Another usage of adders can be found in converting a single-ended signal to differential signals or vice versa. The adder is the primary block for representing the superposition principle of linear system abstraction [37] for analog circuits, along with the filter primitives in the following. The adder and the filter express time-domain and value domain superposition of analog signals.

The scaler is just a special case of adder, having only a single input. The amplifier circuitry is modeled with the scaler by setting the scale parameter as the gain of amplifier. Note that the abovementioned representation for adder, with weighted summation, can be also decomposed into multiple scalers and an adder whose weight parameters all equals to 1.0.

3.2.2 Digital-to-Analog Converter

We state the digital-to-analog converter (DAC) as a specific case of an adder, where the input values only take discrete levels of 0.0 or 1.0. In fact, the model for a DAC can take various representations such as using loop-up table (LUT) representation, using the range of output assuming uniform discretization, or

using the conversion weights corresponding for each input.

While the LUT-style representation can represent most various cases, the problem is that the number of coverage goals explodes to 2^N. On the other hand, specifying only the maximum and minimum range of output generates only two parameters to be covered regardless of the number of inputs. However, specifying only the range assumes the uniform discretization, which has limitation in representing a non-uniform DAC [49]. Thus, we take the linear DAC representation with using the conversion weight parameters.

Many circuit implementations for DAC functionality fits well to the linear model, as the topology of circuit implementation is usually in a serial or parallel array of resistor, capacitors, transistors, with on/off switches. For example, in implementing a current DAC, a parallel array of transistors of different sizes are implemented where the sizes directly map to the weight parameter of DAC. Also, a digitally controlled resistor can be implemented with parallel array or serial string of multiple resistor devices with switches, where the end-to-end transconductance or impedance is the summation of those of the unit devices.

3.2.3 Filter

The analog filter is expressed with the transfer function, where the poles and zeros of transfer function is the parameters. The DC-gain of the filter is set to 1.0 to be exclusive to the scaler. To be physically meaningful, the complex values on the poles are zeros are assumed to always appear with conjugate pairs.

The examples use cases of filters are in the model of loop filters in the phase-locked loops or delay-locked loops, general low-pass filters or linear equalizers,

or data transmitters with finite bandwidths.

The linear filter is widely used in modeling analog circuits even whose operation is not linear in voltage or current domain with the variable-domain translators [37], which demonstrate that the linear system abstraction is valid for the nonlinear circuits such as duty-cycle adjuster, phase interpolator.

Instead of using transfer functions, one can declare the state-variables of the filter and the differential equations, using differentiation or integration operators (e.g., *ddt* and *idt* in Verilog-AMS) with addition and multiplication. However, the transfer function expression (e.g., *laplace_zp* in Verilog-AMS) is more familiar with the usual description format of a filter.

3.2.4 Slicer and Comparator

The analog-to-digital converters (ADCs) compare an analog input to a conversion threshold level and generate Boolean outputs. Following their conditions on changing output in time domain, the ADCs are further categorized into the slicer and the comparator. Both of the slicer and the comparator take the threshold level as the parameter.

The slicer continuously monitors the input and can change its output at any time. One use case of the slicer is in the model of a CML-to-CMOS converter in a clock distribution paths or the integrate-and-fire circuitry [50]. On the other hand, the comparator monitors the input only at the instant that a trigger signal applies, usually given by the edge of a clock signal. The comparator primitives are used in modeling the quantizers in high-speed link receivers or ADCs.

3.3 Custom Computational Blocks

Besides the function primitives, additional blocks are required in describing a circuit model. For example, log or exponential functions (e.g., $f(x) = \log_{20}(x)$, $f(x) = 10^x$) are used to convert signal representations in practice, so that the linear model representations is valid for representing a circuit. Another examples are the inversing or exponentiating of input (e.g., $f(x) = 1/x$, $f(x) = x^A$).

These blocks are also important in describing a model, as the model parameters of real circuit cannot be simple modeled with using linear abstractions only. For example, the bandwidth of an ideal filter consists of an resistor and capacitor can be given as $1/RC$, where the resistance R can be linearly controlled by digital control signals but the value of bandwidth is not. Practical circuits have much more complicated relationship between the device parameter under control (e.g., resistance, capacitance of passive components or transconductance of transistor array) and the model parameter of block (e.g., gain, bandwidth).

Although the custom computation blocks have constants such as the base of log or exponents, those constants are usually not controlled or optimized by a circuit designer to reflect a design intent. Thus, these custom computation blocks are not considered to have important model parameters and excluded in coverage analysis.

3.4 Example Model Description

Figure 3.2 shows the example schematic of a differential amplifier (left). The example circuit employs variable resistors and bias current source, which are controlled by digital signals C_{R1}, C_{R2}, C_{IB}. The functional model (right) of the circuit consists of three DACs, a scaler, a filter and a custom computation block. Figure 3.3 shows the pseudo-code of the model description, written in the syntax of the SystemVerilog.

For this example, model the coverage analysis would be performed on the weight of the three DACs, and gain and the bandwidth value. While gain and bandwidth may vary during simulation time, the coverage analysis is performed for each of actual values appeared as the gain and bandwidth.

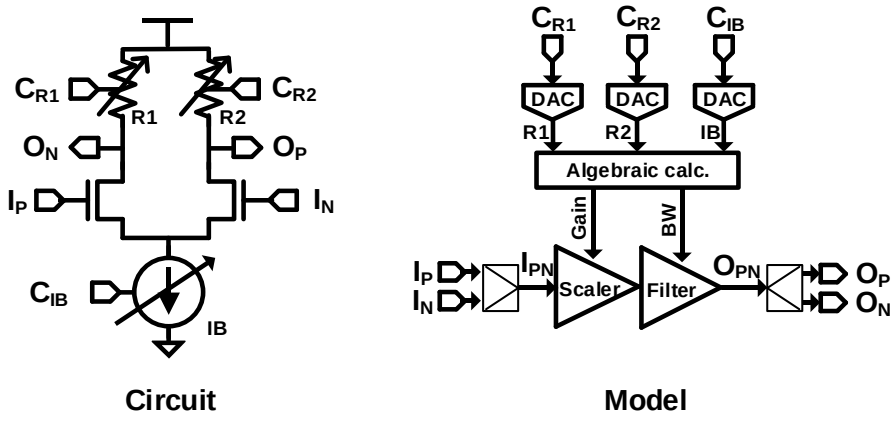


Figure 3.2. Describing the functional model of a differential amplifier with using function primitives.

```
Module Diff_amp(
Input CR1, CR2, CB;
Input IP, IN;
Output ON, OP);

////Paramter-control part
DAC #(.weight(W1)) I0( .out (R1), .in(CR1));
DAC #(.weight(W2)) I1( .out (R2), .in(CR2));
DAC #(.weight(W3)) I2( .out (IB), .in(CIB));
Custom_module I3(.in(CR1, CR2, CR3), .out(Gain, Bw));

/// Main signal processing part
Diff_to_Single I4 (.in(IP, IN), .out(IN_DIFF));
Scaler #(.gain(Gain)) I5 (.in(IN_DIFF), .out(AMP_OUT));
Filter #(.bandwidth(BW)) I6(.in(AMP_OUT), .out(OUT_DIFF));
Single_to_diff I8 (.in(OUT_DIFF), .out(OP, ON));

endmodule
```

Figure 3.3. The pseudo-code of model description for the differential amplifier

Chapter 4

Measuring Waveform Sensitivity

The sensitivity measure translates the notional adequacy of the input signal to a block into a single number. In the literature of analog circuit testing, the sensitivity computation is widely performed to evaluate the effect of a parametric variation in a device (e.g., the threshold voltage of a transistor) in a circuit (e.g., a timing circuitry) to the measurement item (e.g., the delay). However, in functional simulations, the variation due to fabrication process is considered as a secondary effect. Also, to enable the coverage analysis for various analog designs, the sensitivity should be independent to a specific measurement item.

To this end, we derive sensitivity measures that takes an input waveform to an analog block and evaluates the relevancy of the input to the model parameter, without assuming a certain amount of parameter variation or the shape of a waveform.

4.1 Deriving Generic Sensitivity Measure

In deriving the waveform sensitivity, we first evaluate the difference between two distinct analog signals, $y_1(t)$ and $y_2(t)$. Specifically, we measure the difference based on the energy of signals, on a time span of the simulation T_{SIM} :

$$D(y_1(t), y_2(t)) = \frac{\|y_1(t) - y_2(t)\|_2^2}{\|y_1(t)\|_2^2 + \|y_2(t)\|_2^2} \qquad (4.1)$$
$$\text{for } \|y(t)\|_2 = \left(\int_0^{T_{SIM}} y^2(t)dt\right)^{\frac{1}{2}}$$

The measure is symmetric from permutating two signals under comparison and evaluates the relative difference, bounded from 0 to 2. Also, the maximum difference is obtained when $y_2(t) = -y_1(t)$.

Assuming a fixed amount of change in a parameter value, we can get an expression for the output waveform sensitivity to a parameter based on the difference in (4.1).

For example, let a block with N model parameters $\mathbf{p} = [p_1, ..., p_N]$ produces output $y(t|\mathbf{p})$ as the response of its input $x(t)$. Assuming that the k-th parameter p_k is changed by Δ (e.g., $\Delta = 0.5p_k$ for 50% increase), yielding a changed parameter vector $\mathbf{p}'$ and a changed output $y(t|\mathbf{p}')$, the sensitivity S_k can be expressed by using the difference measure between the two output signals:

$$
\begin{aligned}
&S_k(\mathbf{p}, x(t)\,|\,\Delta) \equiv D(y(t\,|\,\mathbf{p}), y(t\,|\,\mathbf{p}')) \\
&\mathbf{p} = [\,p_1, \cdots, p_k, \cdots, p_N\,] \\
&\mathbf{p}' = [\,p_1, \cdots, p_k + \Delta, \cdots, p_N\,]
\end{aligned}
\tag{4.2}
$$

(4.2) can evaluate the sensitivity with an additional computation for $y(t\,|\,\mathbf{p}')$. However, this expression depends on the value of Δ, where we need a proper ground for setting the value of it.

Testing literature [28] refers this as incremental sensitivity, where the process variation provides the grounds for setting the magnitude of parameter deviation. On the other hand, the differential sensitivity is evaluated for infinitesimal change in parameter, taking the limit case of incremental sensitivity. The latter fits more to the functional model simulations that mostly assume idealized parameter values, in contrast to device model parameters that accompanied with variations. Thus, we further remove the conditioning on the value of Δ in the expression.

Revising (4.2), we define sensitivity as the normalized difference at the limit case when Δ goes to zero as illustrated in Figure 4.1. With the sensitivity definition and the relationship of input and output of a block, the sensitivity becomes the function of a model parameter p_k and the input $x(t)$:

$$
S_k(\mathbf{p}, x(t)) \equiv \lim_{\Delta \to 0} \frac{D(y(t\,|\,\mathbf{p}), y(t\,|\,\mathbf{p}'))}{0.5\left(\dfrac{\Delta}{p_k}\right)^2} = {p_k}^2 \frac{\left\| \dfrac{\partial y(t\,|\,\mathbf{p})}{\partial p_k} \right\|_2^2}{\left\| y(t\,|\,\mathbf{p}) \right\|_2^2}
\tag{4.3}
$$

Under this measure, the scales of the signals and the value are normalized by the difference measure and fractional representation of parametric change,

respectively. We further express it using partial derivatives with assuming the model parameters and the input signal are independent to each other. In the following discussion, we use this expression as the sensitivity measure of the adder, DAC, and filter.

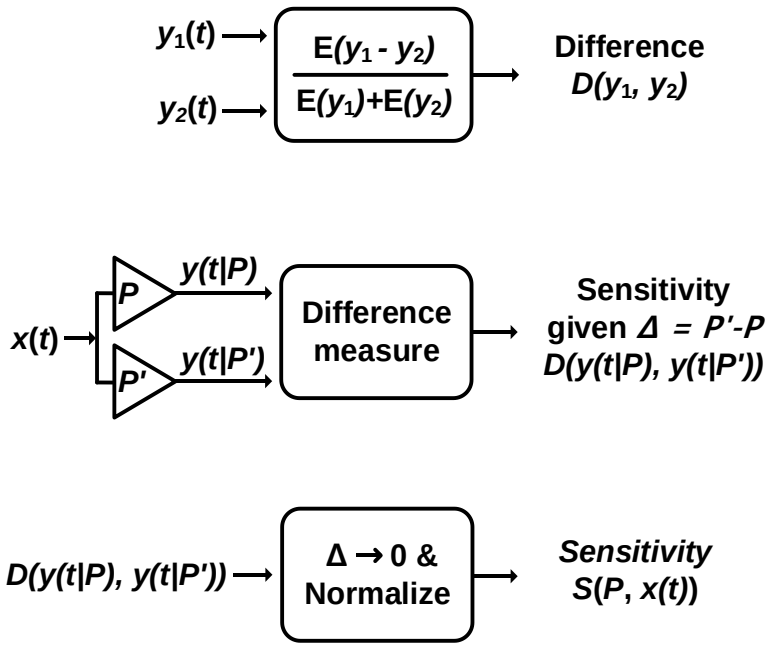


Figure 4.1. Derivation of the sensitivity. Sensitivity measure is a limit case of difference measure when the parameter variation is close to zero.

4.1.1 Adder

An adder computes the output as the weight sum of the inputs $\mathbf{x}(t) = [x_i(t)]$ with weight parameters $\mathbf{w} = [w_i]$, i.e., $y(t) = \sum_{i=1}^{N} w_i x_i(t)$. Here, we assume $N > 1$ as the scaler in the following section addresses $N = 1$. From (4.3), the sensitivity for the k-th weight parameter w_k is given as follows:

$$S_k^{adder}(\mathbf{w}, \mathbf{x}(t)) = w_k^2 \frac{\|x_k(t)\|_2^2}{\|y(t)\|_2^2} \tag{4.4}$$

For the case that all weight parameter have same signs, the sensitivity for a weight parameter is high when the input signal component $x_k(t)$ for the weight w_k is stronger than others. The sensitivity increases as the energy of $x_k(t)$ increases while the energy of total output decreases. In a limit case that single input is dominant, $y(t) \simeq w_k x_k(t)$, the sensitivity converges to 1.0. On the other hand, the different signs of weights may null the output $y(t)$, and the sensitivity can be infinitely large.

Figure 4.3 shows how the sensitivities for a two-input adder are evaluated under different input signal powers, with weights $w_1, w_2 = 1, 2$ and inputs $x_1(t) = \sin(t),\ x_2(t) = A\sin(t)$.

As A increases, starting from zero, the sensitivity for w_2 increases and converges to 1.0 while that for w_1 converges to zero. For a negative value of A, both sensitivities are maximized near $A = -0.5$ and decrease beyond.

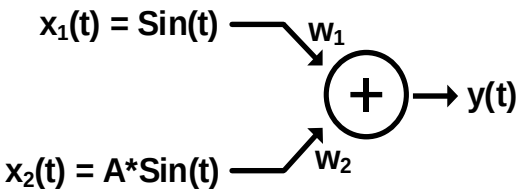


Figure 4.2. Toy example of a two-input adder and the input signals to evaluate the sensitivity function for adder.

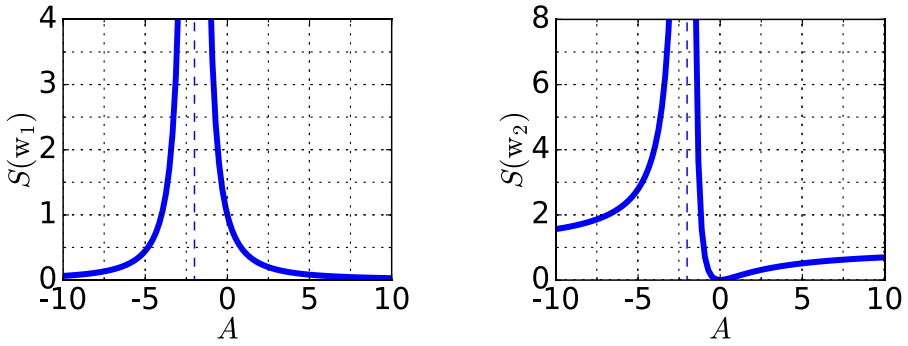


Figure 4.3. The sensitivity analysis for the weights of a two-input adder. As the amplitude of second input changes, the sensitivity varies from zero to infinity or settles to a non-zero level.

4.1.2 Digital-to-Analog Converter

An N-bit DAC produces an output that is the weighted sum of N digital input codes, in similar way to the adder as $y = \sum_{i=1}^{N} w_i x_i$. We consider the DAC primitive as a special case of the analog adder and assign the values for x_i as 0.0 and 1.0 correspond to logic-low/high, respectively.

The sensitivity of the k-th weight w_k for an static input code $\mathbf{x} = [x_k]$ is expressed as same as (4.4):

$$S_k^{DAC}(\mathbf{w},\mathbf{x}) = \frac{w_k^2 x_k^2}{\left(\sum_{i=1}^{N} w_i x_i\right)^2} \tag{4.5}$$

The sensitivity of an all-zero input is defined as zero.

When the k-th input of the DAC is logic low, the sensitivity is just zero. Also, assuming the sign of the weight parameters are same, the sensitivity becomes larger as more of the other input codes have logic-low values.

We extend the sensitivity per an input code to the sensitivity per a sequence of input $\mathbf{X}_{SEQ} = [\mathbf{x}_l]$ that containing L codes, as the average value of the sensitivities for each code in the sequence.

$$S_k^{DAC,SEQ}(\mathbf{w},\mathbf{X}_{SEQ}) = \frac{1}{L}\sum_{l=1}^{L} S_k^{DAC}(\mathbf{w},\mathbf{x}_l) \tag{4.6}$$

Figure 4.5 exemplifies the sensitivity for a three-input DAC with parameters $w_1, w_2, w_3 = 1, 2, 4$, under all possible combinations of input codes. The sensitivity for each parameter ranges from 0 to 1, maximized for the one-hot codes for the input. Further, Figure 4.6 shows the sensitivity measured for a sequence input (4.6) with three cases of different likelihoods for the input code being logic-high. The length of sequence is 20 and the black square marker expresses the logic-high input. The sensitivities also range from 0 to 1, and increase as an input becomes more probable to have logic-high value.

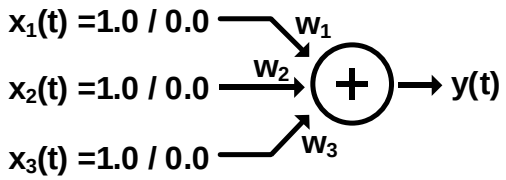


Figure 4.4. Toy example of a 3-input DAC for evaluating the sensitivity of DAC

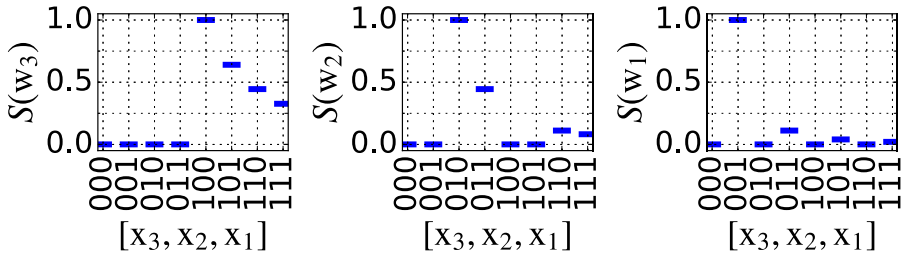


Figure 4.5. Sensitivity for the weights of a three-input DAC, measured per all possible input codes.

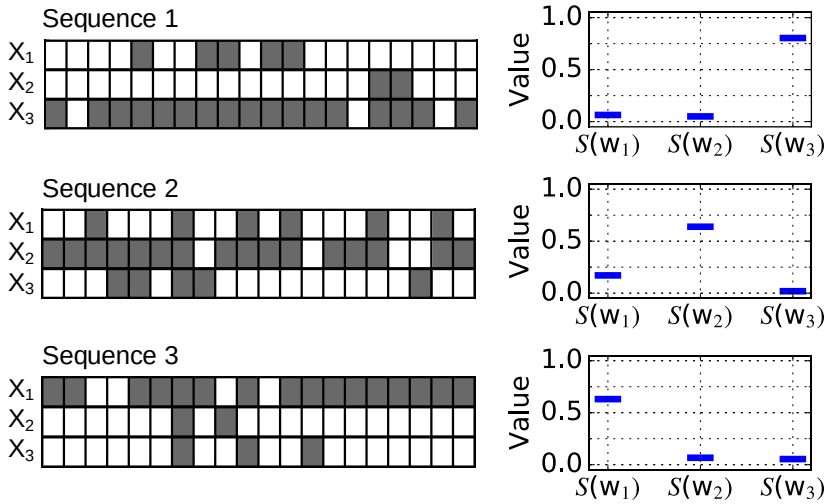


Figure 4.6. Sensitivity for a three-input DAC measured per input sequence. The left figures in each row visualize an input sequence with 10 codes, where the gray/white square represent data "1" and "0," respectively.

4.1.3 Filter

The filter's function is expressed in the s-domain, with the parameters given as poles $\mathbf{p}=[p_i]$ and zeros $\mathbf{z}=[z_j]$ in the transfer function.

$$H(s)=\frac{Y(s)}{X(s)}=\frac{\prod_{j=1}^{N_z}(1+s/z_j)}{\prod_{i=1}^{N_p}(1+s/p_i)} \tag{4.7}$$

Here, N_z and N_p denotes the number of zeros and poles, respectively and we assumed that no duplicated poles or zeros exist and a complex value always appear with its conjugated pair.

Let us first consider the case of real-valued pole (or zero). The sensitivity to a k-th pole (or zero) p_k (or z_k) is derived from directly applying (4.3). In time-domain, the output of linear filter is expressed as $y(t)=x(t)*h(t)$, the convolution of the input $x(t)$ and the impulse response of the filter $h(t)$. The sensitivity is then expressed with time-domain signal representations:

$$S_{pole,k}^{Filter}(\mathbf{p},\mathbf{z},x(t))=\mathrm{p}_k^2\frac{\left\|\frac{\partial(x(t)*h(t))}{\partial p_k}\right\|_2^2}{\left\|x(t)*h(t)\right\|_2^2} \tag{4.8}$$

Assuming that the input $x(t)$ of a filter is independent to the parameter under perturbation, p_k, the partial derivative in the numerator is easily acquired by returning to s-domain. We can change the order of the operator for the Laplace transform, partial derivative and the convolution, finally arriving at the partial derivative of transfer function $H(s)$ with respect to p_k.

As a result, the sensitivity is given by multiplying the original output $Y(s)$

and a high-pass transfer function, revealing that only the high-frequency component of $x(t)$ is only useful in sensitizing the variation in p_k:

$$L\left(\frac{\partial(x(t)*h(t))}{\partial p_k}\right)=L\left(x(t)\right)L\left(\frac{\partial h(t)}{\partial p_k}\right)=X(s)\frac{\partial H(s)}{\partial p_k}$$
$$=X(s)H(s)\frac{s}{p_k(s+p_k)}=Y(s)\frac{s}{p_k(s+p_k)} \tag{4.9}$$

The partial derivative with respect to zeros or duplicated poles also has similar forms. The expression for the sensitivity of a zero z_k only differs the sign from that of the pole:

$$L\left(\frac{\partial(x(t)*h(t))}{\partial z_k}\right)=X(s)\frac{\partial H(s)}{\partial z_k}=Y(s)\frac{-s}{z_k(s+z_k)} \tag{4.10}$$

In general, a pole p_k can have multiplicity of N_0 greater than 1. In this case, the expression for sensitivity is derived similarly to the previous cases:

$$\frac{\partial}{\partial p}\left(\frac{1}{\left(1+s/p\right)^{N_0}}\right)=\frac{1}{\left(1+s/p\right)^{N_0}}\left(N_0\frac{s}{p(s+p)}\right) \tag{4.11}$$

Specifically, the transfer function containing two complex-conjugated poles can be expressed in two forms, with the real and imaginary part of the poles $\{P_R, P_I\}$ or the natural frequency ω_0 and damping factor ζ:

$$\frac{1}{\left(1+\frac{s}{p_R+jp_I}\right)\left(1+\frac{s}{p_R-jp_I}\right)}=\frac{1}{1+\zeta\frac{s}{\omega_0}+\frac{s^2}{\omega_0^2}} \tag{4.12}$$

In this case, the parameter sensitivity can be defined from the partial derivative with respect to either $\{p_R, p_I\}$ or $\{\omega_0, \zeta\}$.

Figure 4.8 shows the sensitivity of a first-order or second-order filter under sinusoidal input $\sin(\omega t)$ of varying frequency. The first row shows the sensitivity of a 1-st order filter with $p = 10^9$ Rad/s and a second order filter with two real pols $p_1 = 10^9$, $p_2 = 10^{10}$ Rad/s. The second and third rows show the sensitivity of 2-nd order filter with $\omega_0 = 10^9$ Rad/s and three values of ζ's. For all the cases, the sensitivity transits from a low to high level as ω becomes larger than the poles. Specifically, the sensitivity for ζ is high only when ω is near the system's pole, indicating that the change in damping factor is most critical for the resonant case. Also, for different values of ζ, the sensitivities of ω_0 and ζ, or P_R and P_I show conflicting trends, where the increase in the sensitivity for one parameter is accompanied by the decrease in the sensitivity of the other parameter.

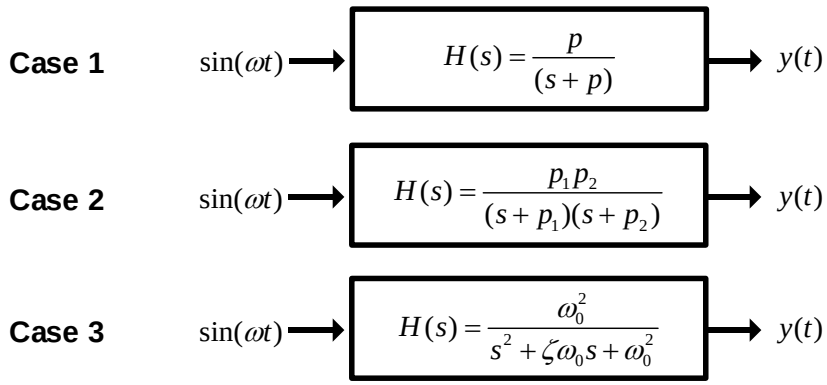


Figure 4.7. Examples of filters taking sinusoidal inputs: a first-order filter with pole (p), second-order filter with two real poles ({ p_1, p_2 }), a second-order filter with complex conjugate poles, ({ p_R, p_I } or { ω_0, ζ }).

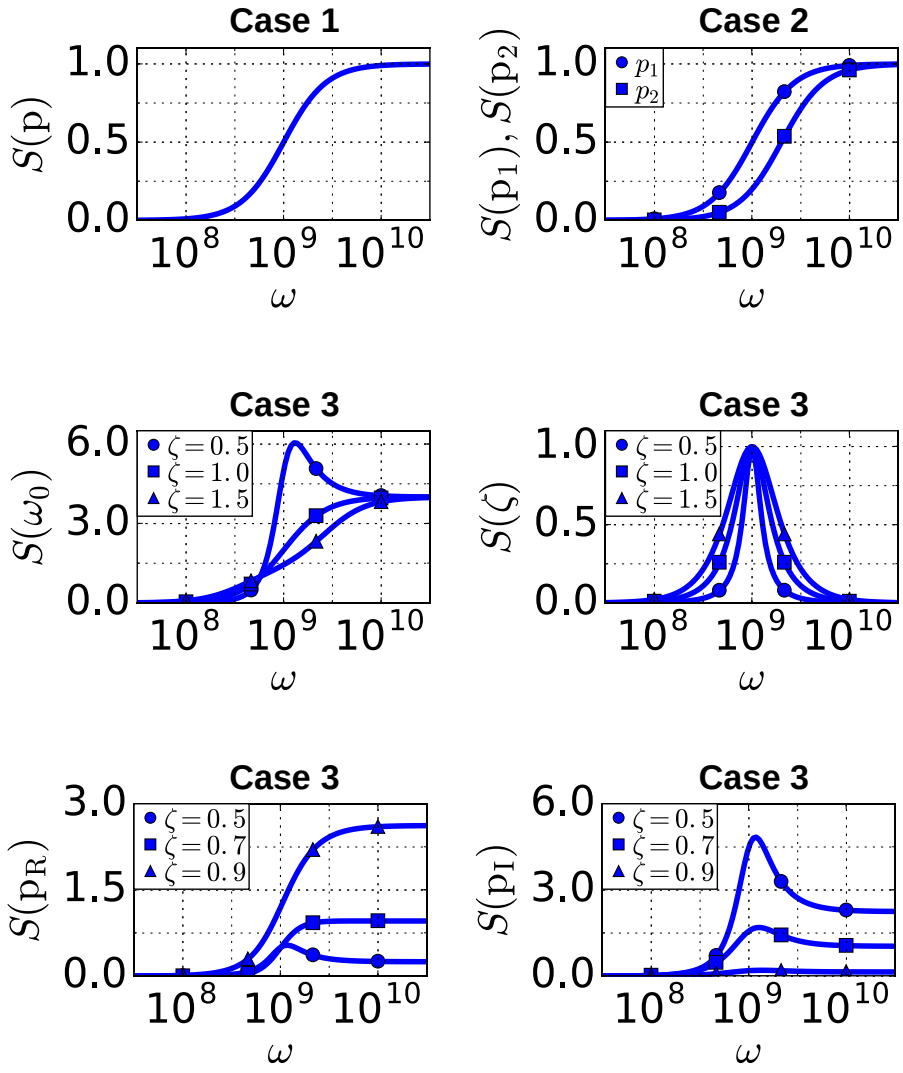


Figure 4.8. Sensitivity for the three examples in Figure 4.7. For the second-order filter examples, the damping factor is also swept. The regions in frequency axis for high sensitivity and low sensitivity are clearly distinguished.

4.2 Sensitivity of Scaler

The scaler's function is equivalent to that of a one-input adder, with a single parameter w. However, directly applying the sensitivity expression (4.3) produces a constant value regardless of the change in the input. As a result, the amount of relevancy of input signals cannot be differentiated and we modify the sensitivity function discussed so far. We modify denominator of (4.3) with the energy of maximum of output signal during the simulation time span.

$$S_k(\mathbf{p}, x(t)) = p_k^2 \frac{\int_0^{T_{SIM}} \left(\frac{\partial y(t)}{\partial p_k} \right)^2 dt}{\int_0^{T_{SIM}} \left(\max_t (y(t)) \right)^2 dt} \tag{4.13}$$

Similar to the original sensitivity measure function, (4.13) is also devised to normalize the scale of the signals and the parameters. Applying this expression for scaler, the sensitivity is defined as measuring the amount of non-zero content in the input under the constraint on the range of the input signal:

$$S^{Scaler}(w, x(t)) = \frac{\int_0^{T_{SIM}} x^2(t) dt}{\max_t (x^2(t)) T_{SIM}} \tag{4.14}$$

Note that the sensitivity for a scaler is independent of both the parameter value and remains constant under the scale of the input signal (e.g., $\tilde{x}(t) = kx(t)$.

As an example, Figure 4.9 visualizes four example input signals for a scaler and the corresponding sensitivity values. The sample inputs are rectangular, sinusoidal, triangular, and squared triangular functions of time, which all have

the period of 1.0 and range from -1.0 to 1.0. Among them, the rectangular wave exhibits largest value of the sensitivity while the squared triangular function shows the smallest value of sensitivity.

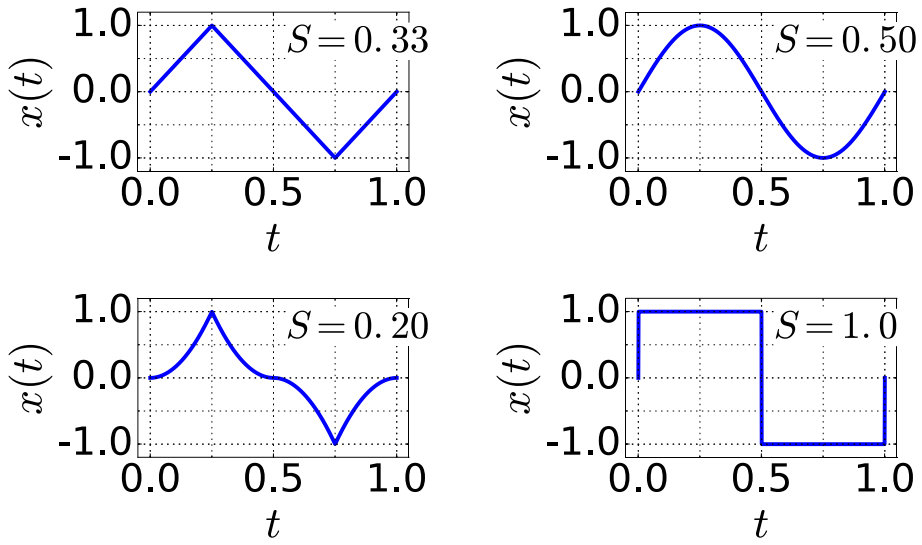


Figure 4.9. Examples input signals for a scaler and the corresponding sensitivity values (*S*). Note that the sensitivity value does not depend on the scale parameter of the scaler.

4.3 Sensitivity of Slicer for Transition Timing of Output

The slicer asynchronously produces a digital output by comparing the continuous input signal to a threshold parameter V_{TH} as $y(t) = \mathrm{sgn}(x(t) - V_{TH})$. Here, $\mathrm{sgn}(\cdot)$ denotes the sign of its argument, being the value of either +1 or -1. Let $x(t)$ crosses V_{TH} at time 0 with falling and crosses V_{TH} again at time T with rising, then the sensitivity is defined as:

$$S^{Slicer}(V_{TH}, x(t)) \equiv \frac{\partial T}{\partial V_{TH}} \frac{H}{T} = \frac{H}{T}\left(\left. \frac{dx}{dt} \right|_{@t=T} \right)^{-1}$$
$$H = \frac{\int_0^T (V_{TH} - x(t))dt}{T} \tag{4.15}$$

This sensitivity measure evaluates the variation in the output toggle timing due to variations in V_{TH}, which is inversely proportional to the instantaneous slope of input $dx(t)/dt$ at a time instant T when the input crosses the threshold (i.e., $x(T) = V_{TH}$).

As int the previous sensitivity expression for linear systems or scaler, the sensitivity for the slicer is normalized under scaling in time and the difference from V_{TH}. When the input is scaled in time, by $\tilde{x}(t) = x(t / t_0)$ with some constant t_0, or in the amount of the difference from V_{TH}, by $\tilde{x}(t) - \mathrm{V}_{TH} = V_0(x(t) - \mathrm{V}_{TH})$ with some constant V_0, the sensitivity does not change. Figure 4.10 explains the meaning of sensitivity function and the normalization term (H/T).

Figure 4.11 shows example waveforms of four input signals to a slicer with

$V_{TH} = 0$. The input signals are exponentially decaying one with dc-offset, or ramp-up, sinusoidal, triangular function of time, all crossing V_{TH} at time 0.0 and 1.0. Among them, the exponentially decaying function shows largest sensitivity of 0.63 while the sinusoidal function shows the smallest sensitivity of 0.21.

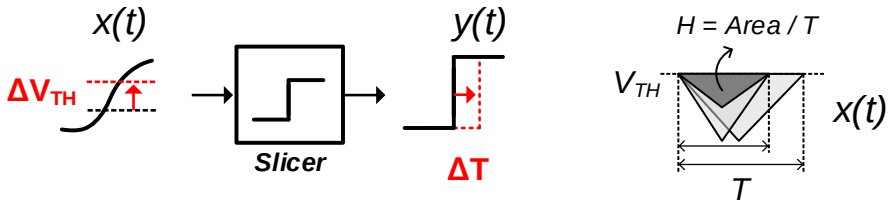


Figure 4.10. The sensitivity for slicer is defined on the timing variation on the output logic signal. Further, the sensitivity provides normalization by calculating the average height (H).

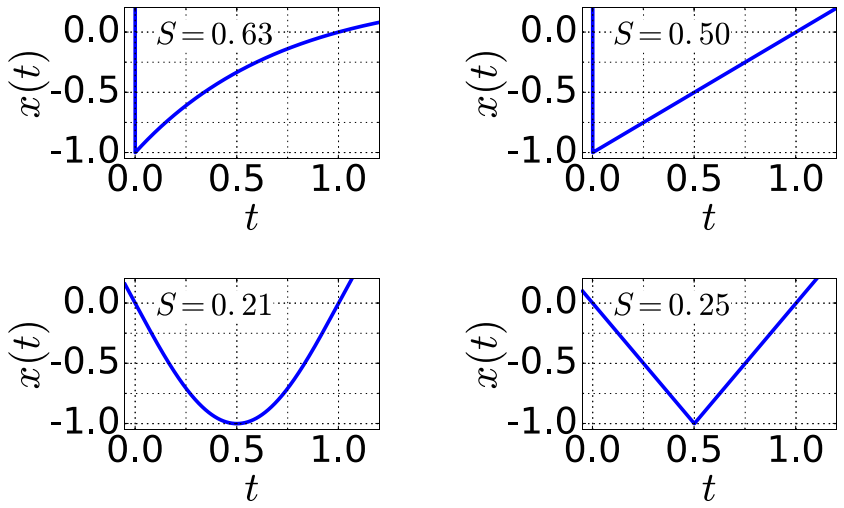


Figure 4.11. Example inputs for a slicer with $V_{TH} = 0$ and corresponding sensitivity values (*S*).

4.4 Sensitivity of Comparator for Expected Value of Output

The comparator functions similarly to the slicer with a threshold parameter V_{TH} ; however, it compares a sampled value of input to the threshold, instead of continuous comparison. That is, the output is determined as $y(t_i) = \mathrm{sgn}(x(t_i) - V_{TH})$ for discrete time instants t_i, $i = 1, \cdots, N$ at which input is sampled. Here, the value of $\mathrm{sgn}(\cdot)$ takes ± 1, following the sign of its input. While the comparator actually takes two inputs $x(t)$ and t_i , we define sensitivity for the $x(t)$ by taking the partial derivative of the expected value of the output samples:

$$\begin{aligned} E[y] &= -\int_{-\infty}^{V_{TH}} P_X(x)dx + \int_{V_{TH}}^{\infty} P_X(x)dx \\ S^{Comp.}(\mathrm{V}_{TH}, x(t)) &\equiv \frac{\dfrac{\partial E[y]}{2\partial V_{TH}}}{\sigma_x^{-1}} = \sigma_x P_X(x = V_{TH}) \end{aligned} \tag{4.16}$$

Here, $P_X(x)$ denotes the probability density function (PDF) of input samples. Under this sensitivity, an input with more samples near the V_{TH} is evaluated as a better one for sensitizing the variation in V_{TH} . Also, The expression includes normalization with the inverse of standard deviation, σ_x, to be not affected by scaling of the input range (e.g. $P_{\tilde{X}}(\tilde{x}) = 1/k\, P_X(x/k)$).

Figure 4.13 shows four example PDFs for the input of comparator and the corresponding sensitivity values, assuming V_{TH} = 0.5. Upper left and lower left each show two beta distribution with shape parameters (0.2, 0.2) and (1.0, 1.5). Upper right and lower right each show a triangular probability distribution and

a normal distribution $N(0.5, 0.2^2)$. Among them, the sensitivity of triangular PDF is the largest, while that of the convex beta distribution is the smallest.

In practice, the input PDF for a comparator is estimated from the discrete samples of the input, but the way of data interpolation or smoothing affects the value for $P_X(x = V_{TH})$. Thus, to avoid over- or under-smoothing, the existing correlation between input data samples should be considered in proper way.

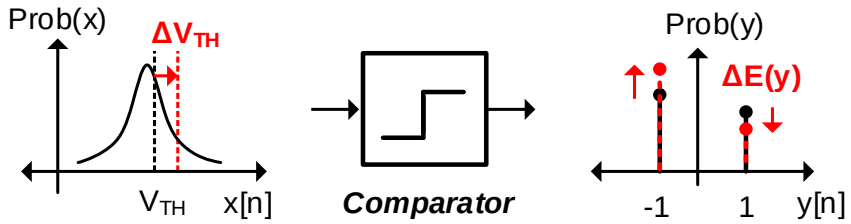


Figure 4.12. The sensitivity of comparator is defined basically on the variation of expected value of output due to the variation in the conversion threshold.

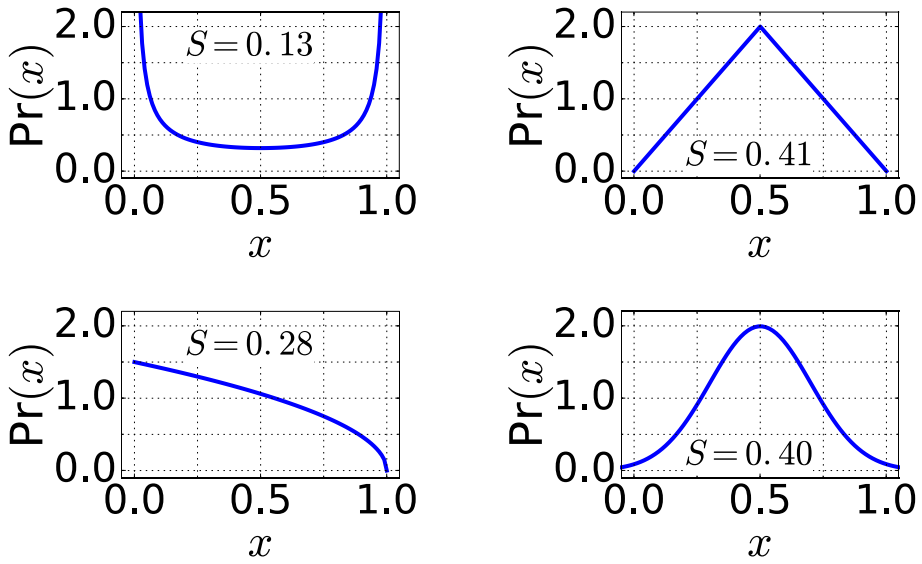


Figure 4.13. Example inputs of a comparator and corresponding sensitivity values (*S*) for V_{TH} = 0.5.

4.5 Summary on Physical Meaning of Sensitivity Functions

Table 4.1 summarizes the physical meaning of sensitivity function for each primitive. For function primitives related with linear abstractions, such as Adder, DAC, Filter, Scaler, the sensitivity functions measure the relative energy of input. Following the functionality of each primitives, the energies are normalized regarding the energies of other inputs, or the distribution on the frequency domain, or the maximum value. For slicer and comparator, the sensitivity functions measure the instantaneous slope of input signal or the amount of concentration of input near the conversion threshold. The common characteristics of the proposed sensitivity measures are the normalization of signal scales and the scale of parameters.

Table 4.1. Summary on the physical meaning of sensitivity measure functions for each function primitives.

Type	Meaning of Sensitivity
Adder	Energy of an input signal, relative to that of others
DAC	Energy of an input code, relative to that of others
Filter	Spectral distribution of input power, relative to its maximum
Scaler	Energy of the input signal, relative to its maximum
Slicer	Slope of input at crossing V_{TH}, relative to the aspect ratio
Comparator	Density of input samples near V_{TH}, relative to the spread

4.6 Sensitivity Measure for Multiple Blocks

The sensitivity measures derived in this section extends to the case of blocks connected in serial. For example, Figure 4.14 illustrates two blocks, each with parameter p and q. The functions of the first and the second block are stated as $y(t) = f(x(t)|p)$ and $z(t) = g(y(t)|q)$, respectively. Then, we can define the sensitivity of the signal $z(t)$ with respect to the change in p (not q in constrast to the discussion so far) extending the sensitivity defined in (4.3).

$$S_z(p, x(t)) = p^2 \frac{\left\| \frac{\partial g(y(t)|q)}{\partial p} \right\|_2^2}{\left\| g(y(t)|q) \right\|_2^2} = p^2 \frac{\left\| \frac{\partial g(f(x(t)|p)|q)}{\partial p} \right\|_2^2}{\left\| g(f(x(t)|p)|q) \right\|_2^2} \qquad (4.17)$$

Unfortunately, the extended definition of this sensitivity is not always directly acquired from the originally defined sensitivities for the two blocks, $S_y(p, x(t))$ and $S_z(q, y(t))$. Instead, (4.17) involves to derive the change in $z(t)$ with respect to the change in $y(t)$, in contrast to the previous cases that only address the souce of change is in the value of a parameter.

We discuss three cases, where the second block being a scaler, an adder, or a filter. For simplicity, p and q are assumed to be independent. When the second block is a scaler, giving $z(t) = qy(t) = qf(x(t)|p)$, then $S_z(p, x(t))$ equals to $S_y(p, x(t))$ as in (4.18). Second case is when $z(t) = \sum q_i y_i(t)$, assuming $y(t)$ is a vector-valued signal. If only the k-th component of $y(t)$ depends on $x(t)$, $y_k(t) = f(x(t)|p)$, then $S_z(p, x(t))$ becomes scaled value of $S_y(p, x(t))$ as (4.19). The third case is that the second block being a filter, giving $z(t) = y(t) * h(t|q)$, where the operator $*$ denotes the convolution and

$h(t|q)$ is the impulse response of the filter. Then, $S_z(p, x(t))$ is derived as (4.20), with the convolution of $\partial y(t)/\partial p$ and $h(t|q)$, since the effect of perturbation on $y(t)$ due to p cumulates on $z(t)$ over time.

$$p^2 \frac{q^2 \left\| \frac{\partial(f(x(t)|p))}{\partial p} \right\|_2^2}{\|z(t)\|_2^2} = p^2 \frac{q^2 \left\| \frac{\partial y(t)}{\partial p} \right\|_2^2}{\|z(t)\|_2^2} = p^2 \frac{\left\| \frac{\partial y(t)}{\partial p} \right\|_2^2}{\|y(t)\|_2^2} \tag{4.18}$$

$$S_y(p, x(t)) \times \frac{\|q_k y_k(t)\|_2^2}{\|z(t)\|_2^2} \tag{4.19}$$

$$p^2 \frac{\left\| \frac{\partial y(t)}{\partial p} * h(t|q) \right\|_2^2}{\|z(t)\|_2^2} = S_y(p, x(t)) \times \frac{\|y(t)\|_2^2}{\|z(t)\|_2^2} \times \frac{\left\| \frac{\partial y(t)}{\partial p} * h(t|q) \right\|_2^2}{\left\| \frac{\partial y(t)}{\partial p} \right\|_2^2} \tag{4.20}$$

In summary, calculating sensitivity for multiple blocks needs more computations than simply multiplying the sensitivities of single blocks derived through Chapter 4.1- 4.4. However, the expressions are still in closed form and the coverage metric can be derived in the same framework as the case of using sensitivity functions for individual blocks.

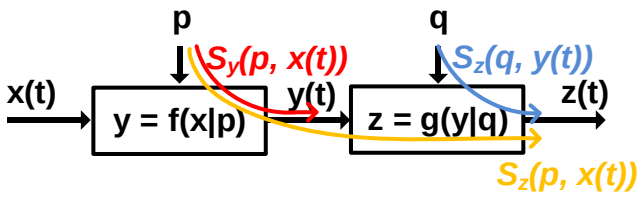


Figure 4.14. Sensitivity defined for multiple blocks.

Chapter 5

Decision Threshold of Sensitivity

After deriving the sensitivity measure, we further need a decision threshold value to decide the coverage for a parameter. In deriving the threshold, the challenge arises since the sensitivity takes continuous value on infinite range. While a verification engineer can manually set the threshold value with design expertise, it requires to track the actual ranges for sensitivity values on every block instances of the model.

Thus, to make the coverage analysis applicable for different designs and various simulation scenarios, we need to have a generic threshold value that is valid for various instances of a primitive block. In this chapter, we derive the framework for deriving such threshold based on the information theory and present representative analysis for the function primitives introduced in the Chapter 2.

5.1 Information-Maximizing Criteria

We model the coverage decision procedure in (2.1) as the Bernoulli random process, based on defining an input space for each primitive.

The input space is a *hypothetic ensemble* of possible input signals for a function primitive. We parameterize an input signal by a real-valued random vector $\mathbf{p}_\mathbf{X}$, which is defined on a set $\Omega_{\mathbf{P}_\mathbf{X}} \subset \mathbb{R}^N$, with a probability distribution described by a function $f(\mathbf{p}_\mathbf{X})$.

$$\begin{aligned} &x(t) = x(t|\mathbf{p}_\mathbf{X}) \in \Omega_x, \\ &\text{for } \mathbf{p}_\mathbf{X} \in \Omega_{\mathbf{p}_\mathbf{X}},\ \Pr(\mathbf{p}_\mathbf{X}) = f(\mathbf{p}_\mathbf{X}) \end{aligned} \tag{5.1}$$

Subsequently, the coverage decision process is to receive an input $x(t)$ and to produce an on/off result through measuring the sensitivity provided by the input $x(t)$ and comparing it against a threshold. (Figure 5.1 (a)).

Here, the sensitivity measure is deterministic function and the threshold is a given constant. On the other hand, the input is determined by the random variable $\mathbf{p}_\mathbf{X}$, thus making the coverage decision process as a random process. Figure 5.1 (b) illustrates this, where the randomness arises from the input signal space.

With this formalization, the optimum threshold θ_{Opt} for coverage decision (2.1) is defined as the value that maximizes the information entropy of the process (Figure 5.1 (c)). Since the coverage decision yield on/off results, the binary entropy function for the Bernoulli process can be used here.

A Bernoulli random process is characterized by the probability of the output being true, which we denote as k. Provided the input space definition and

sensitivity measure for a block, k only depends on the value of the decision threshold. Subsequently, we obtain θ_{Opt} by maximizing the binary entropy function; this corresponds to make $k(\theta_{Opt})$ to be 0.5, or equivalently to find the median of the possible sensitivity values.

$$\begin{aligned} k(\theta) &= \Pr(\mathrm{S(p, x(t))} > \theta) \\ \theta_{Opt} &= \arg\max_{\theta}(-k\log_2(k) - (1-k)\log_2(1-k)) \end{aligned} \tag{5.2}$$

The analysis is summarized operationally as defining the input space for each function primitives and obtaining the optimal decision threshold. Once the formalization is performed for each function primitive, it is applied for each instance of a model without further manual adjustments.

The following subsections present the representative analysis for general cases. For each function primitive, we present the actual definition for the input signal space and derive the decision threshold, which turns out to be a closed-form expression of parameters.

A verification engineer can tailor the input space definitions for specific designs, instead of using universal input spaces presented for each function primitives. In literature, [32] also presents to construct behavioral input signal space for specific designs examples, based on assuming the shape of signal functions and extract performance metrics from the simulation result.

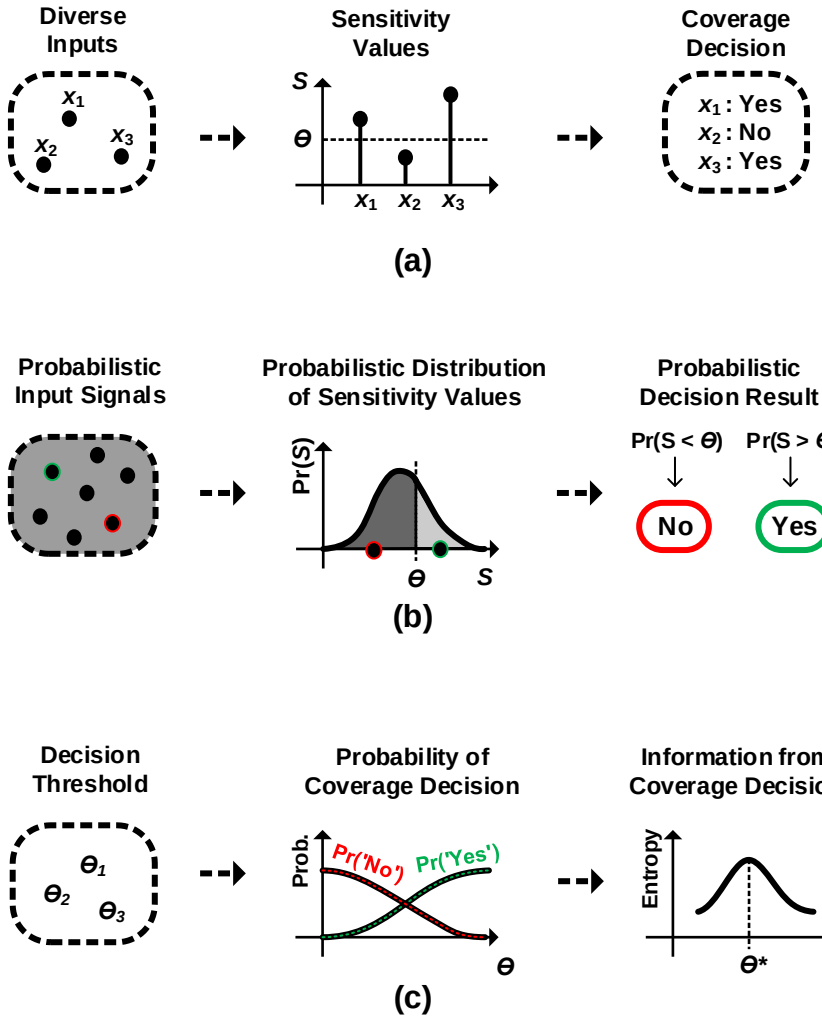


Figure 5.1. (a) Coverage is determined by whether the sensitivity (S) is greater than a threshold (Θ). (b) From probabilistic input signals, the outcome of coverage decision becomes a random process. (c) The optimum threshold is that maximizes the information entropy of coverage decision

5.2 DC Input Signal Space for Adder

As a starting point, the input space of an adder is defined as DC signals on an N -dimensional unit circle with uniform probability distribution.

$$\begin{aligned} &x_i(t) = x_i,\ \textstyle\sum_{i=1}^{N} x_i^2 < 1 \\ &\Pr(\mathbf{x}) \sim \text{uniform on } \textstyle\sum_{i=1}^{N} x_i^2 < 1 \end{aligned} \tag{5.3}$$

With this definition of the input space, the optimal decision threshold for the sensitivity of the k-th weight of adder is given as:

$$\theta_{Opt,k}^{Adder} = \frac{w_k^{\,2}}{\sum_{i=1}^{N} \mathrm{w}_i^2} \tag{5.4}$$

The optimality of the threshold is derived by showing a probability relation on the input space:

$$\Pr(\mathrm{S}_k^{Adder} > \theta_{Opt,k}^{Adder}) = \Pr((\mathbf{a}_+^{\mathrm{T}}\mathbf{x})(\mathbf{a}_-^{\mathrm{T}}\mathbf{x}) > 0) = 0.5 \tag{5.5}$$

Here, $\mathbf{a}_{\pm} = (\mathbf{w}^{\mathrm{T}}\mathbf{w})\mathbf{e}_k \pm (\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}})\mathbf{w}$ for the k-th standard basis vector $\mathbf{e}_k$, $\mathbf{x}$ and $\mathbf{w}$ are the vector of input signals and of the weight of the adder, respectively.

Then, two hyperplanes defined as $\mathbf{a}_+^{\mathrm{T}}\mathbf{x} = 0$ and $\mathbf{a}_-^{\mathrm{T}}\mathbf{x} = 0$ divide the input signal space into four equal volumes because $\mathbf{a}_+^{\mathrm{T}}\mathbf{a}_- = 0$. Thus, the probability value is 0.5, satisfying our condition for the optimal threshold. Figure 5.2 illustrates the input space and division in 3-dimensional case.

The suggested threshold maintains its optimality for other definition of input signals, where the geometrical division can be applied. The region on which signal parameters are defined or the different probability distribution can be

defined.

For example, any DC input spaces that are symmetric about the origin, such as $\max|x_i| \le k$ or $\sum_{i=1}^{N}|x_i| \le k$ with any positive real k can also use the same threshold. Furthermore, the threshold is optimal for certain classes of non-DC signals, such as $x_i(t) = k_i \sin(\omega t)$ when k_i 's are symmetrically distribution about the origin.

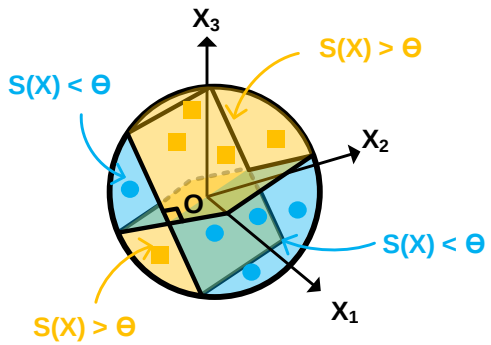


Figure 5.2. The input space of an adder is defined by a N-dimensional hypersphere. The optimum threshold corresponds to the two orthogonal hyperplanes that evenly divides the hypersphere.

5.3 DC Input Signal Space for Digital-to-Analog Converter

We assume that for an N-bit DAC, its input code is distributed uniformly over 2^N possible values. Subsequently, the optimal threshold for the k-th weight is given as:

$$\theta_{Opt,k}^{DAC} = \frac{w_k{}^2}{\left(\sum_{i=1}^{N} w_i\right)^2} \tag{5.6}$$

Being similar to the adder, the optimality is derived from the probability relation on the N-dimensional hypercube $[0,1]^N$ of which the vertices represent the input of the DAC:

$$\Pr(S_k^{DAC} > \theta_{Opt,k}^{DAC}) = \Pr((\mathbf{b}_+^{\mathrm{T}}\mathbf{x})(\mathbf{b}_-^{\mathrm{T}}\mathbf{x}) > 0) = 0.5 \tag{5.7}$$

Here, $\mathbf{b}_{\pm} = (\mathbf{1}^{\mathrm{T}}\mathbf{w})\mathbf{e}_k \pm \mathbf{w}$ and $\mathbf{1}$ denotes the all-one vector and we assume that all w_i's are of the same sign. Subsequently, two hyperplanes $\mathbf{b}_-^{\mathrm{T}}\mathbf{x} = 0$ and $\mathbf{b}_+^{\mathrm{T}}\mathbf{x} = 0$ each evenly divides or does not intersects to the hypercube, respectively. Thus, the vertices of the hypercube are divided evenly into two groups, with designating the all-one/all-zero inputs as the covering and not-covering inputs. In fact, this decision criterion yields equivalent result to check whether the sensitivity is zero or not. Figure 5.4 illustrates the input space and division of 3-dimensional case.

A more stringent decision threshold can be found by restricting the input space that produces only nonzero output sensitivities. In this case, the threshold is computed by obtaining the median of all 2^N-1 non-zero sensitivity values:

$$\theta_{Opt,k,Nonzero}^{DAC} = median(\{S_k^{DAC}(\mathbf{w},\mathbf{x}) \mid x_k = 1\}) \tag{5.8}$$

While the expression for this threshold is not a closed-form as in the previous cases, we can easily compute the actual threshold value by computing and sorting all the 2^N-1 non-zero cases in short time.

We also derive the optimal decision threshold for the sensitivity on the sequence of inputs. Here we assume that each code in the sequence is independently and identically distributed. If the sequence is sufficiently long, we can approximate that the sensitivity follows a normal distribution due to the central limit theorem. Subsequently, the optimal decision threshold corresponds to the mean of the distribution, which is the average of all the 2^N values of sensitivities for a code as (5.9). Since $S_k^{DAC}(\mathbf{w},\mathbf{x})$ is given in a closed-form and the number of all possible cases are finite, one can further derive the expression for $\theta_{Opt,k}^{DAC,SEQ}$ also in a closed-form.

$$\theta_{Opt,k}^{DAC,SEQ} = \frac{1}{2^N} \sum_{\text{"all possible } \mathbf{x}\text{"}} \left(S_k^{DAC}(\mathbf{w},\mathbf{x})\right) \tag{5.9}$$

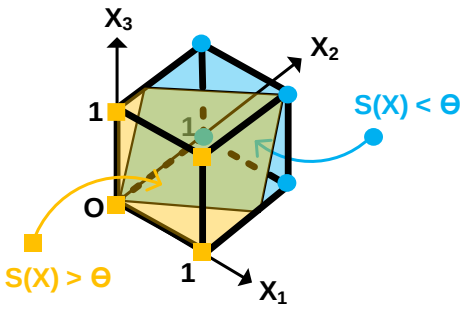


Figure 5.3. The input space of a DAC is the vertices of a N-dimensional hypercube. The optimum threshold corresponds to the hyperplane that evenly divides the hypercube.

5.4 Frequency-Domain Input Signal Space for Filter

We utilize Parseval's theorem in defining the input signal space for a filter in the frequency domain. The time-domain integration for sensitivity is converted to that of the frequency domain and further approximated to sum the spectral components on discretized frequency intervals:

$$\frac{\int(\partial y(t)/\partial p)^2 dt}{\int y^2(t)dt} = \frac{\int(x(t)*\partial h(t)/\partial p)^2 dt}{\int(x(t)*h(t))^2 dt} = \frac{\int|X(j\omega)|^2 |\partial H(j\omega)/\partial p|^2 d\omega}{\int|X(j\omega)|^2 |H(j\omega)|^2 d\omega} \cong \frac{\sum x_k h_k'}{\sum x_k h_k} \tag{5.10}$$

with h_k', h_k, x_k 's that approximate the square of spectral component of the transfer function, its partial derivative, and the input signal by their average value on each frequency interval $[\omega_k, \omega_{k+1}]$, respectively.

To utilize this approximation, the input space is defined on discretized intervals in the frequency domain, in which the power spectrums are constant. For each frequency interval, let the power content follows a uniform distribution on interval [0, A] and be independent of each frequency intervals.

$$|X(j\omega)|^2 = x_k \text{ on } \omega_k < \omega < \omega_{k+1}$$
$$x_k \sim unif(0, A),\ 0 < \omega < \omega_{MAX} \tag{5.11}$$

Further, we constraint the maximum frequency by a sufficiently large value ω_{MAX}, over which components of the sensitivity change little. The spectrum on the negative axis for the frequency is assumed to be symmetric to the positive

axis, on which we define the input signal.

Following (5.11), the optimum decision threshold for filter is then derived from the following probability relation.

$$\Pr\left(\frac{\sum x_k h'_k}{\sum x_k h_k} > \frac{\sum h'_k}{\sum h_k}\right) = \Pr(\mathbf{a}^{\mathrm{T}}\mathbf{x} > 0) = 0.5 \tag{5.12}$$

where $\mathbf{a}^{\mathrm{T}} = (\mathbf{h}^{\mathrm{T}}\mathbf{1})\mathbf{h}'^{\mathrm{T}} - (\mathbf{h}'^{\mathrm{T}}\mathbf{1})\mathbf{h}^{\mathrm{T}}$ is defined from $\mathbf{h}' = [h'_k]$ and $\mathbf{h} = [h_k]$. Because $\mathbf{a}$ defines a hyperplane containing the origin and $\mathbf{1}$, it cuts a hypercube $[0, \mathrm{A}]^N$ evenly, making the probability 0.5. Figure 5.4 illustrates this. The threshold corresponds to the case where all the spectral components of input signal have the same values, which is the frequency spectrum of the impulse function. Finally, the optimum decision threshold is defined from the sensitivity of impulse response of the system:

$$\theta_{Opt,p}^{Filter} = p^2 \frac{\left\|\partial h(t)/\partial p\right\|_2^2}{\left\|h(t)\right\|_2^2} \tag{5.13}$$

Assuming the form of transfer function to a certain type of filter, the expression of the threshold can be further revised as the function of transfer function parameter (p). For example, the impulse response of a 1-pole filter with unit DC-gain is given as $h(t) = pe^{-pt}$. Then, the expression of (5.13) is rearranged, with $\partial h(t)/\partial p = (1 - pt)e^{-pt}$, as:

$$\frac{\left\|\partial h(t)/\partial p\right\|_2^2}{\left\|h(t)\right\|_2^2} = \frac{\int_{T_1}^{T_2}\left(e^{-2pt} - 2pte^{-2pt} + p^2t^2e^{-2pt}\right)dt}{\int_{T_1}^{T_2}\left(p^2e^{-2pt}\right)dt} \tag{5.14}$$

Assuming the simulation is sufficiently long and the signal is observed on the entire simulation time, T_1, T_2 can be set to $0, \infty$. Then, the integral of

(5.14) evaluates to $1/\left(2p^2\right)$, and thus the $\theta_{Opt,p}^{Filter}$ evaluates to 0.5. Referring the example of Figure 4.8 with sinusoidal excitations, this threshold value corresponds to the case when input freqency equals to the pole of filter.

For the filters of higher number of poles, the impluse response can be written as $h(t)=\sum_k\left(r_k t^{n_k} e^{-p_k t}\right)$ by partial-fraction expansion. Also T_1, T_2 can be arbitrary numbers satisfying $T_1 < T_2$. Still, the integral can be expressed in closed-form as the indefinite integral of (5.15) assuming p_i's are real-valued.

$$\int t^n e^{-pt} dt = \frac{-1}{p^{n+1}} e^{-at} \sum_{k=0}^{n} \left(\frac{n!}{(n-k)!} (pt)^{n-k} \right) \tag{5.15}$$

Again for $T_1, T_2 = \{0, \infty\}$, (5.15) is further simplified as $n!a^{-(n+1)}$. For complex-conjugate pairs of p_i's, the threshold can be also derived by finding closed-form expresison of $\int t^n e^{-at}\cos(-\omega t)dt$ or $\int t^n e^{-at}\sin(-\omega t)dt$.

Note that we standardized the form of unit transfer function not as $H(s)=1/(s+p)$ but as $H(s)=p/(s+p)$ for the filters (Table 3.1). Since the DC-gain of the former also depends on p and thus being sensitive to the change in p, the former type of transfer function becomes sensitive on the entire range of frequency domain. In order to make the sensitivity on p only related with the poles, the latter expressions of transfer function is used.

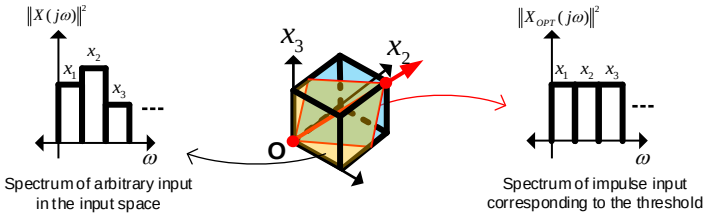


Figure 5.4. In the input space for a filter, the optimal decision threshold coincides to the input with constant power on the frequency spectrum.

5.5 Time-domain Input Signal Space for Scaler

The input space scaler consists of a piecewise-constant signal on uniformly discretized time intervals. Further, the probability distribution of the signal values for each time duration is provided as a uniform distribution constrained in the range defined by an arbitrary positive real number C.

$$\begin{aligned} &x(t) = x_i \text{ for } t_i < t < t_{i+1}, \\ &x_i \sim unif(-C, C),\ i = 1, \cdots, N \end{aligned} \tag{5.16}$$

Here, the overall range of t is given as the total time space of simulation, or the provided time window for which coverage monitoring is performed. With this definition of input space and the sensitivity measure for the slicer, the optimal decision threshold is given as:

$$\theta_{Opt}^{Slicer} = \frac{E[x_i^2]}{C^2} = \frac{1}{3} \tag{5.17}$$

The threshold value coincides with the sensitivity given by ramp or sawtooth or triangular input to the scaler. In fact, the threshold corresponds to any time-domain signal whose samples at discrete time points have uniform distribution on the interval $[-C, C]$.

Note that the threshold does not depend on the specific value of the scale parameter of the scaler, in contrast to the previous cases where the expressions for threshold are given as the function of the parameters. Also, as in the previous cases, the threshold is valid for any positive C due to the normalization

provided by (4.14).

The optimality of the threshold is derived by showing the following probability relation:

$$\Pr(S^{Scaler} > \theta_{Opt}^{Scaler}) = \Pr\left(\frac{\sum_{i=1}^{N} x_i^2}{N} > E[x_i^2] \right) = 0.5 \qquad (5.18)$$

Here we assume each x_i in (5.16) is independently and identically distributed. Then, central limit theorem states that the left-hand side of the inequality ($\sum_{i=1}^{N} x_i^2 / N$) will follow Gaussian distribution for a large N . Also, as the theorem states that the expectation $E[x_i^2]$ approximately equals to the mean of $\sum_{i=1}^{N} x_i^2 / N$. Since the mean and the median of a Gaussian random variable are equal, the expression means the probability of the sample of a random variable is larger than its median value. Thus, the probability of the inequality evaluates to 0.5.

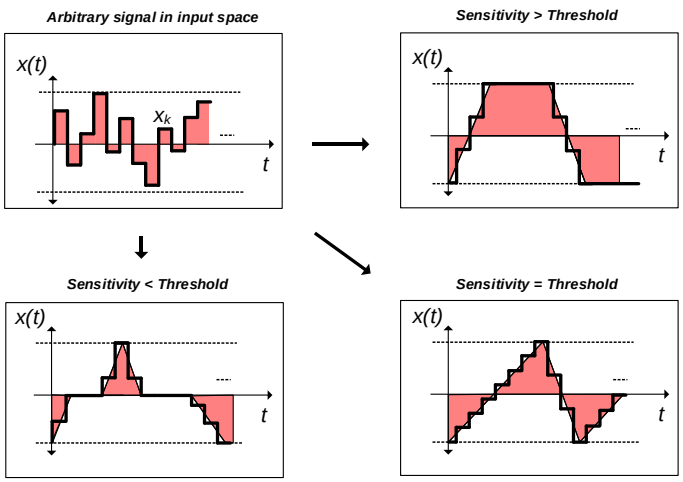


Figure 5.5. The input signal space for scaler and the case corresponding for decision threshold.

5.6 Varying-Curvature Input Signal Space for Slicer

We construct the input space for a slicer with a parameterized basis function, which is defined on $0 < t < T_0$. The value of basis function is $V_{TH} - V_0$ at time zero. Starting at this point, the function is a monotonically increasing signal with various curvatures and cross V_{TH} at time T_0:

$$x(t) = V_{TH} + V_0\left(-1 + \left(\frac{t}{T_0}\right)^N\right), \qquad (5.19)$$
$$N \sim \text{unif}(\alpha, \beta) \text{ for } \alpha, \beta, V_0, T_0 > 0$$

Here, α and β define the range of curvature and $\{V_0, T_0\}$ define the scale in the time/value domain. To maintain the shape of signals as monotonically increasing ones, α, β, V_0, T_0 are assumed to be positive. The curvature of signal is determined by a positive real number N, which is assumed to follow uniform probability distribution. Figure 5.6 shows an example signal traces for N ranges from 10 to $1/10$, assuming that $V_{TH} = 0$, $V_0 = 1$, $T_0 = 1$.

We derive optimal threshold by applying the expression of sensitivity measure for the slicer, (4.15), to the input signal of slice in (5.19). Thanks to the normalization terms, the sensitivity becomes independent to V_0, T_0, V_{TH} and becomes a monotonic function of N:

$$S^{Slicer} = \frac{\frac{1}{T_0^2}\int_0^{T_0}\left(V_{TH} - x(t)\right)dt}{\left.\frac{dx(t)}{dt}\right|_{T_0}} = \frac{\frac{V_0}{T_0^2}\left(T_0 - \frac{T_0^{N+1}}{(N+1)T_0^N}\right)}{V_0\frac{NT_0^{N-1}}{T_0^N}} = \frac{1}{N+1} \qquad (5.20)$$

As we assumed that N is uniformly distributed on the range $[\alpha,\beta]$, we can find the optimum decision threshold as the sensitivity for the midpoint of the range $[\alpha,\beta]$, as $\left(0.5(\alpha+\beta)+1\right)^{-1}$. However, it is highly affected by the value of α and β.

As an alternative, we define the optimal decision threshold as the mean of the two sensitivity values of N being α or β, as:

$$\theta_{Opt}^{Slicer} = 0.5\left(\frac{1}{\alpha+1}+\frac{1}{\beta+1}\right) \tag{5.21}$$

For the symmetric case that $\alpha\beta = 1$, the optimal decision threshold becomes 0.5, which coincides to the sensitivity given by applying linear ramping-up signal to the slicer. Also, for the case of $\alpha \ll 1 \ll \beta$, the threshold approximately evaluated as 0.5.

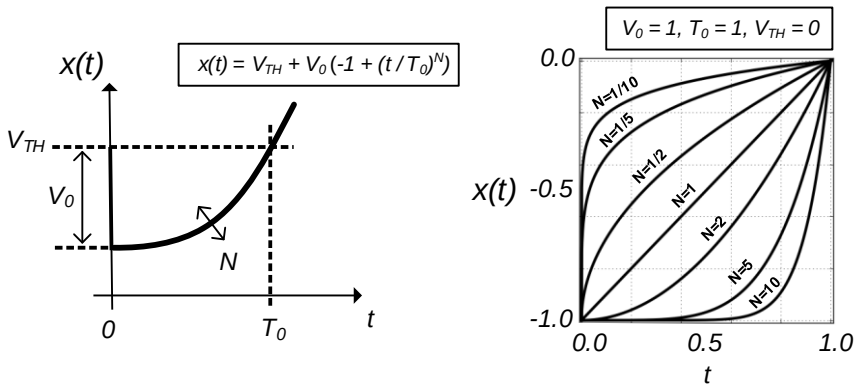


Figure 5.6. The input signal space for the slicer is defined from the basis function, which has randomized parameter that controls the curvature of the signal.

5.7 Input Probability Distribution Space of Comparator

The input space for a comparator consists of probability density functions, whose density values are piece-wisely constant on the discretized intervals for the input x. Assuming that x ranges on a finite interval $[x_{\min}, x_{\max}]$ which uniformly is discretized into N subintervals, the density value on each interval is defined:

$$\begin{aligned} &P_X(x) = p_i \text{ on } x_i < x < x_{i+1} \text{ for } i = 0,1,\cdots,N \\ &x_{\min} = x_0, x_N = x_{\max} \\ &p_i \sim \text{unif}(0, p_{\max}) \text{ and } \sum_{i=1}^{N} p_i = 1 \end{aligned} \tag{5.22}$$

Here we can assume that N and $p_{\max}$ are sufficiently large. Also, we further assume that $V_{TH} = 0$ and the range of x is symmetric to V_{TH} as $x_{\min} = -1.0$, $x_{\max} = 1.0$ since we defined the sensitivity measure function to normalize the standard deviation of x. Then, the optimal decision threshold is calculated by generating random samples in the input space and searching the median of the sensitivity values. From the experiment, the optimal decision threshold is found as $\theta_{Opt}^{Comp.} \cong 0.29$.

Referring the examples in Chapter 4.4, the value of the decision threshold states that most probability distributions for x centered at the V_{TH} in concave form, being a unimodal distribution, would cover the V_{TH}.

Figure 5.8 shows the pseudo-code for the numerical experiments. In the procedure, first a set of positive random numbers are generated following the probability distribution for p_i in (5.22), as a single instance of input for the

comparator. Then the sensitivity value is calculated following (4.16), with computing sample standard deviation and setting $P_X(x=0=V_{TH})$ from the array of generated random values. For example, when total 1000 random numbers are generated, with N_{BINS} being 1000, the 500-th values is chosen as the value of $P_X(x=0)$. This random generation of input PDF for comparator and calculation of the sensitivity value are repeated N_{MC} times, which approximates the distribution of sensitivity values under the definition of input space in (5.22). From the array of N_{MC} sensitivity values, the median value is found as the sample of $\theta_{Opt}^{Comp.}$. Finally, the entire procedures are repeated N_{REPEAT} times and the final result for $\theta_{Opt}^{Comp.}$ is given as the mean of the N_{REPEAT} sample values of $\theta_{Opt}^{Comp.}$.

Figure 5.9(a) shows the histogram of $\theta_{Opt}^{Comp.}$ values found under N_{MC}=10000, N_{REPEAT} = 10000, N_{BINS} = 1000 and Figure 5.9(b) lists the sample standard deviation of $\theta_{Opt}^{Comp.}$ values for different N_{MC} values.

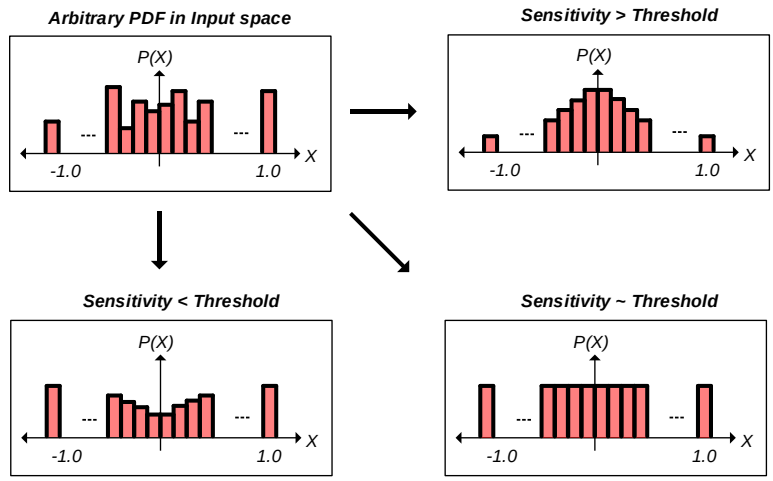


Figure 5.7. Input space for a comparator is piecewise constant probability density function on a discretized finite interval.

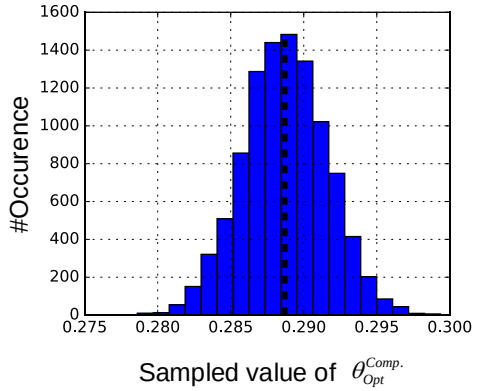
(1) *Generate positive random numbers of size* N_{BINS} *(e.g., 1000) with uniform distribution on a range and scale the numbers so that the sum of numbers equals 1.0.*

(2) *Calculate the sample standard deviation and the sensitivity following the definition of sensitivity* $S = \sigma_X^{-1} P_X(x = 0 = V_{TH})$.

(3) *Repeat (1)-(2)* N_{MC} *times to approximate the distribution of sensitivity values and find the median values of sensitivity from the sorted list*

(4) *Repeat (3)* N_{REPEAT} *times and find the mean of sensitivity values*

Figure 5.8. Pseudo-code of numerical experiments for finding optimum threshold values for the sensitivity of the comparator.



(a)

N_{MC}	Mean of $\theta_{Opt}^{Comp.}$	S.D. of $\theta_{Opt}^{Comp.}$
100	0.289237	0.0125
1000	0.288951	0.00390
10000	0.288934	0.00130

(b)

Figure 5.9. The distribution of optimal decision threshold value for comparator found by repetitive numerical experiments.

5.8 Characterizing Instance-Specific Input Signal Space

While we presented general input space for each function primitives and optimal threshold values, the input space may not proper for specific instances of a primitive blocks in a certain design. Following the structure of the design or the condition of system's input for simulation, an internal block can take input signals that can be different from the presented input signal spaces.

For those cases, the sensitivity function will still valid but the presented threshold values may be inappropriate, being too pessimistic or optimistic with respect to the actual formation of input signals in typical simulations.

Following the idea of [32], we can construct an input signal space for a known design, from existing simulation traces that are assumed to be sufficient. In [32], the feature values of input signals are extracted for a charge-pump in a phased-locked loop (PLL), with properly assuming the shape of waveforms since the input of charge-pump comes from the phase-detector that generates pulse signals.

Then, for example, we can collect the samples of sensitivity values for the charge-pump block and can find the optimum decision threshold for the sample distribution of sensitivity values. After finding the threshold, we can annotate it back to the charge-pump block in the model. In contrast to the deciding coverage only from identifying the type of block and parameter values, this annotation is for specific instance block and can be embedded in the model for later simulations, such as verifying the PLL with other blocks in more higher

level of system integration or verifying a similar PLL with a new testbench. For those cases, the instance-specific threshold value can be used for self-checking of the coverage. In general, the approach can be applied to the instances of the function primitive blocks discussed in this dissertation.

Since assuming a 'golden' simulation is dangerous in verification, this approach should be carefully used to mitigate the false positive or false negative decision on the coverage. This approach would be more beneficial when the controllability of the input signals for an internal block is well understood and the mismatch to the generic input signal space is related to the operation of the entire DUV.

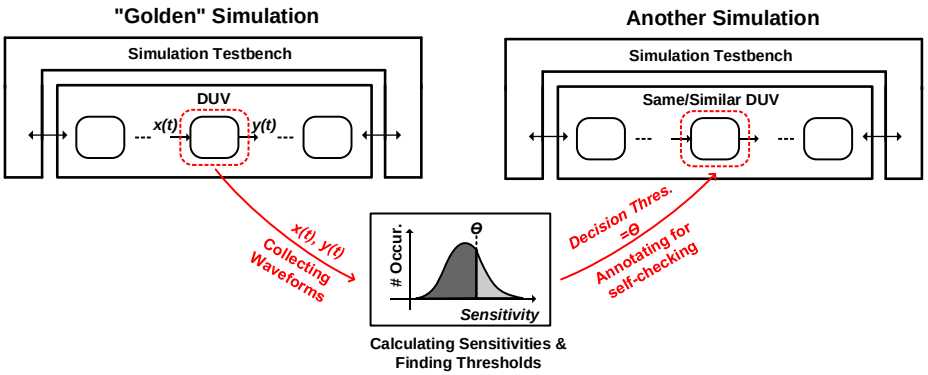


Figure 5.10. Extracting the sensitivity values from a 'golden' simulation and finding the threshold values empirically. The threshold values can be further annotated to the specific instance block in DUV.

Chapter 6

Experimental Results for Block and System Examples

In this Chapter, we exhibit how the proposed sensitivities are evaluated and coverage is decided for typical simulations scenarios of real-world block examples. The block examples are the decision feedback equalizer (DFE), feedforward equalizer (FFE), continuous-time linear equalizer (CTLE), integrate-and-fire cell, and two-bit delta-sigma modulator (DSM); each highlights the results for specific function primitives in themselves. Further, a system-level example of an analog frontend of RF receiver shows the practical usage in equivalence checking between two system models.

6.1 Implementation of Model Simulation and Coverage Analysis

We used an analog behavioral simulator XMODEL [33] to perform the behavioral simulation and implement sensitivity computations. Using XMODEL, we implemented custom blocks for signal energy computations and simulated them together with the normal blocks for the original analog functional model. For the remaining computations required for coverage analysis, Python scripts were produced to post-process the simulation result. Also, digital HDL models in some of the examples are also simulated together with analog functional models in XMODEL.

In example models, there are time-varying parameters as well as static parameters. Example of time-varying parameters is the gain of a variable gain amplifier, whose gain parameter is determined by digital control signals during simulation. In this case, we assumed that the value of gain changes significantly slower than the main input of the amplifier. Subsequently, the coverage was measured for the time interval during which a single gain value is being held, out of the total simulation time span. For a single simulation that exhibits different gain values, the coverage for each of appearing gain values are decided.

Figure 6.1 shows the implementation of coverage analysis for the adder primitive and filter primitive. The computation of signal energies for sensitivity measurements are performed inside the original simulation model, where we extended original signal computation models. Referring the figure, the block *Ecalc* calculates the integral of the square input signal from the start of

simulation time to the current simulation time. That is, the output $y(t)$ of Ecalc for its input $x(t)$ is given as $y(t) = \int_0^t x^2(\tau) d\tau$, which is implemented by built-in primitive of XMODEL, namely *square* primitive and *integ* primitive.

After the a simulation finishes, the parameter sensitivity is calculated in the Python script by reading the signal waveforms and sampling the value of energy signals (the output of Ecalc) at two time points, T_1, T_2, which are the start and the end of the observation time window for simulation results. The default values of T_1, T_2 are the 0 and end of total simulation time and manually chosen. The other parts of computations in coverage decision procedures are straightforward, which are to find threshold values from parameters of the target primitive block and comparing the sensitivity and the threshold value.

Figure 6.1 only shows the case of adder and the filter. For other primitives, such as scaler, DAC, slicer, comparator, most computations are performed in the Python script. The computations inside post-processing script demand to read digital or analog signal values from simulation results. Specifically, the computation for slicer involves signal derivatives, and crossing time of analog signals. These signal processing functions are implemented based on Python extension libraries provided by XMODEL.

Most of the signal processing required for the proposed coverage is not dependent to specific simulators, so the implementation is also possible on various other simulators or model description languages, such as the real number models in SystemVerilog [1]. Yet, the detailed implementations depend on the way of expressing signals in each simulators and models.

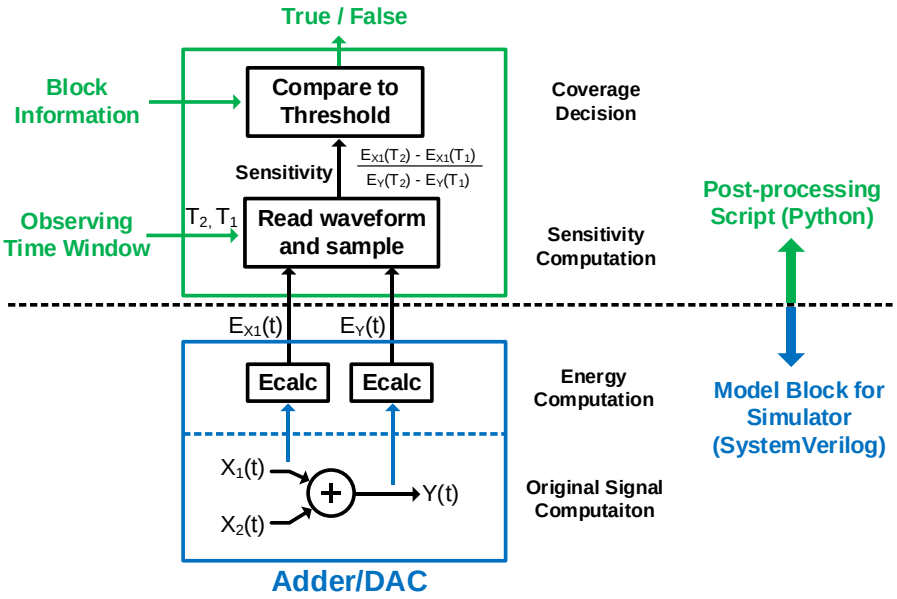


Figure 6.1. The implementation of coverage decision framework for adder/DAC using modified blocks in simulator and post-processing script.

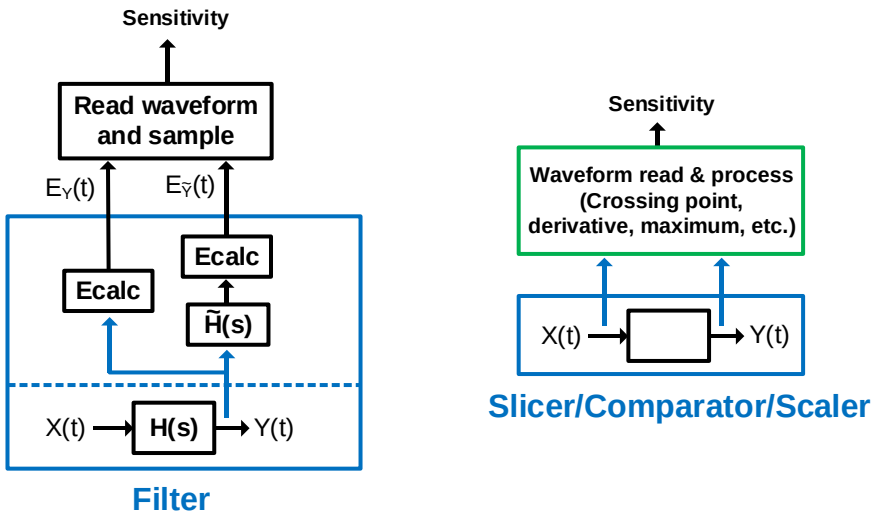


Figure 6.2. The part of implementation of coverage decision framework for Filter, Slicer, Comparator, Scaler

6.2 Decision-Feedback Equalizer and Feedforward Equalizer

The decision-feedback equalizer and the feedforward equalizer are chosen to show the numerical result of applying the parameter coverage analysis on the instances of adder and DAC primitive.

Figure 6.3(a) shows the block diagram of a three-tap DFE model, which includes the adder primitives. The model consists of two adders, a comparator, a shift register, and three one-bit DACs that are assumed to map logic "1" and "0" to 1.0 and –1.0, respectively.

The weights of adders are $\{ w_A, w_B \}$ = {1.0, –1.0} and $\{ w_1, w_2, w_3 \}$ = {0.24, –0.045, –0.089}. The comparator has threshold 0.0 and feedback 1-bit DACs produces +/–1.0 output value. The average value of V_{IN} is assumed to zero.

The DFE is simulated with binary PRBS data on V_{IN} that are assumed to appear through a channel with low-pass characteristic. The sensitivities are measured on each period of the data duration, divided by the sampling clock of the data comparator. Figure 6.5(a) shows the sensitivity for the three-input adder that uses discrete levels following the combination of the data decision value. The trend is identical for the three weights because their input values can take only 1.0 or –1.0.

The second adder contains input signal sets that comprise more continuous set of values. Figure 6.5(b) shows the histogram of the observed sensitivity values for the second adder. The dashed lines in figures represent the decision thresholds of each parameters derived in Chapter 5.

The results indicate that constant bit sequences are the best inputs for sensitizing the parametric variation for the three-input adder. This is because the constant bit sequence yields the least amount of inter-symbol-interference, minimizing the energy of signal V_{FB} for equalization. For the primary adder, w_A has larger sensitivity than w_B, as the signal energy of V_{FB} is usually smaller than that of V_{IN}.

Figure 6.3(b) shows a block diagram of a three-input FFE, containing a three-input DAC and two-input adder. The model implements an FFE with tap coefficients of {–0.2, 0.7, –0.1}, by mapping it to the function primitives of this study with a DAC that have weights of equal sign.

The weights for DAC and adder are $\{ w_3, w_2, w_1 \}$ = {0.2, 0.7, 0.1} and $\{ w_A, w_B \}$={1, 1} and V_{DC} = –0.3. The input D_{IN} is given as logic-valued signals and the shift register deserializes three consecutive digital bits to the DAC.

Figure 6.4 shows the sensitivity for the weights of DAC and adder under simulating all possible data patterns for the FFE. As the FFE contains three taps, and eight types of bit patterns completely exhibit all possible sensitivity values in the blocks.

Similar to the case of the DFE, data pattern that generates minimum output power shows larger sensitivity values. In covering all the five parameters, a minimum set of stimuli is found as {100, 010, 001} from the results. Dashed lines represent the coverage decision thresholds derived in (5.6).

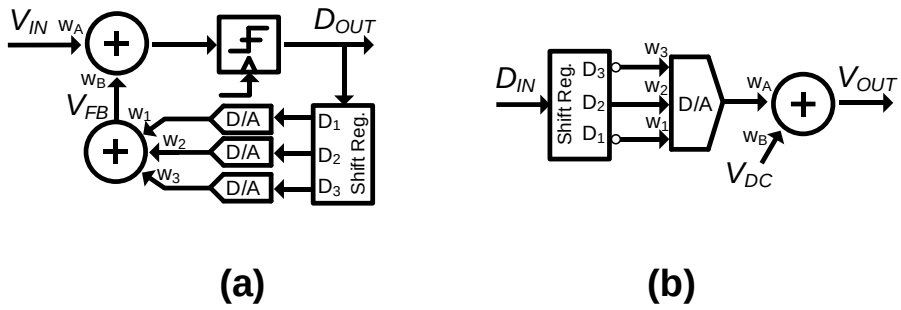


Figure 6.3. Block diagrams of a three-tap decision feedback equalizer and a three-tap feedforward equalizer.

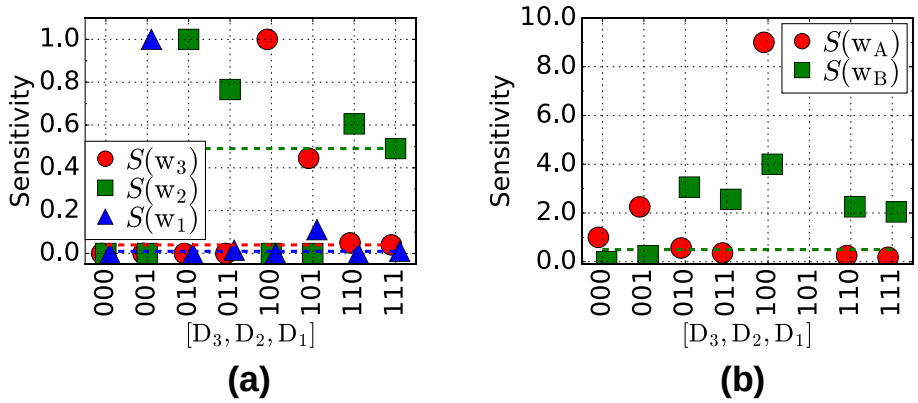


Figure 6.4. Sensitivity of the parameters (a) in the adder, and (b) in the DAC of the FFE example with respect to three consecutive values of input data pattern.

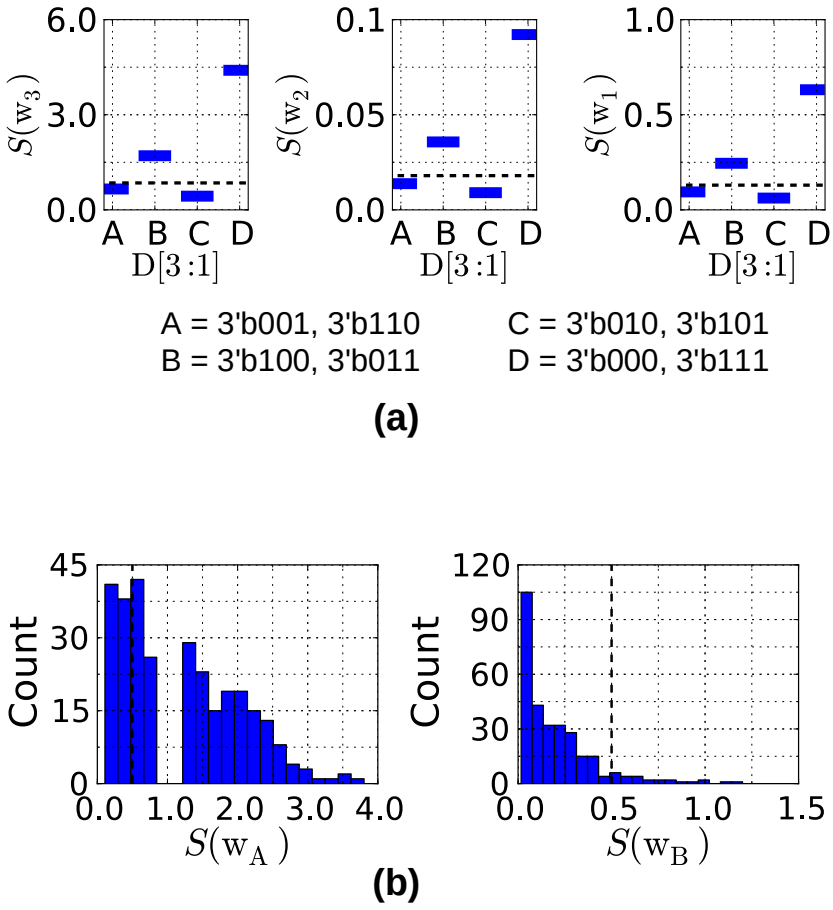


Figure 6.5. Sensitivity of the adders in the DFE example, (a) for the primary adder (two-input) under PRBS data pattern, and (b) for the feedback adder (three-input) under all possible feedback data patterns.

6.3 Continuous-Time Linear Equalizer and Biquad Low-Pass Filters

As block examples for filters, a second order CTLE and a fourth order low-pass filter (LPF) are selected, which are typically used in wireless/wireline receivers. The CTLE is modeled as a single filter, having a zero (z) and two poles (p_1, p_2) at $0.15, 0.5, 1.0 \times 2\pi \times 10^9$ Rad/s and the DC gain is 0.66. The LPF is modeled as two cascading second-order low pass filters, with model parameters of $\omega_0 = 2\pi \times 10^6$ Rad/s and DC gain is set to 1.0 for both stages. ζ is 0.9 and 0.5 for the first and second stage, respectively.

The simulation input for the CTLE are two types. The first one is to apply a single-tone sinusoidal input with varying frequency (ω_{in}). The second one is more realistic signals in data transmission with varying data rate and four different data patterns. The levels of original transmitted data are +/–1.0 and that signal is assumed to be passed through a lossy channel. The channel is also modeled as a linear low-pass filter with transfer function parameters, where loss is –24dB at 4 GHz.

Figure 6.7 shows the sensitivity for the CTLE simulated under two scenarios. For sinusoidal inputs, the frequency ω_{in} should be approximately larger than the zero or pole values (Figure 6.7(a)). However, for real data patterns, the 1010 pattern with data rate larger than 2 Gbps can only cover all of the parameters (Figure 6.7(b)). This is because the inputs for the CTLE pass through the lossy channel; thus, the PRBS or 1110 pattern input is shaped as a dominant content in low-frequency region, when the input signal arrives at the CTLE.

Figure 6.8 shows the result of simulating the biquad LPFs with a sinusoidal input or the sum of the baseband modulated signal of bandwidth 1 MHz, with binary phase-shift keying scheme, and an interferer signal with the same baseband bandwidth but offset at 10 MHz. For a sinusoidal input, the results are the same as those in Figure 6.7(a), where the poles are covered when the input frequencies are larger than ~1 MHz. Meanwhile, ζ 's are covered only for frequencies near 1 MHz. For the input with modulated data signals, Figure 6.8(b) shows that the parameters for the first-stage LPF are covered when the power of the interferer becomes larger than ~10x of the primary channel. Also, Fig. 17(b) shows that $\omega_{0,2nd}$ and ζ_{2nd} do not exhibit sufficiently large sensitivities and are not covered in the simulation scenario. Dashed lines in Figure 6.7 and Figure 6.8 represent the coverage decision thresholds for filter's parameters derived in (5.13).

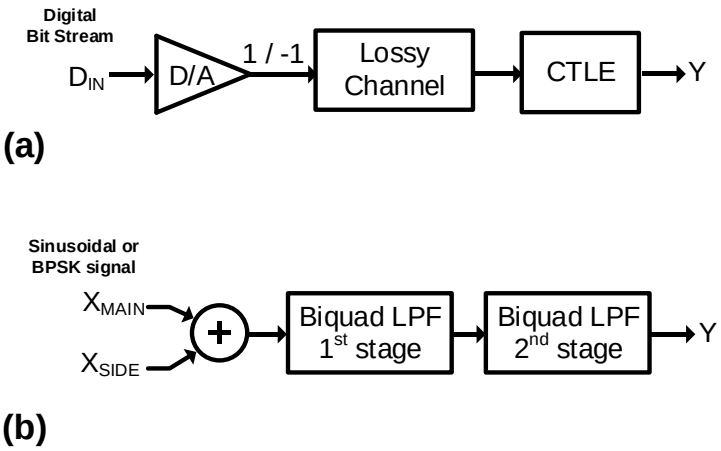


Figure 6.6. Simulation scenarios for (a) a continuous-time linear equalizer and (b) a fourth order low-pass filter.

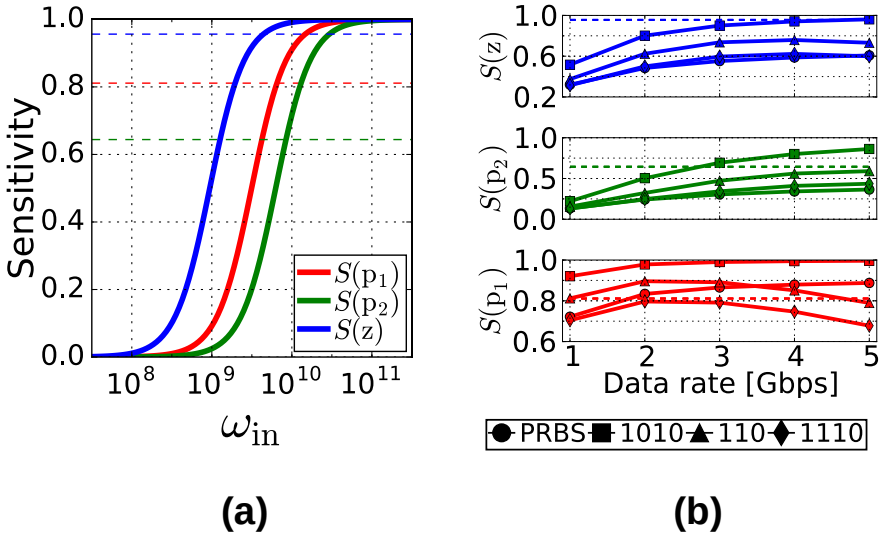


Figure 6.7. Parameter sensitivity of CTLE under (a) single-tone sinusoidal input with varying frequency and (b) bit stream of four patterns with varying data rate.

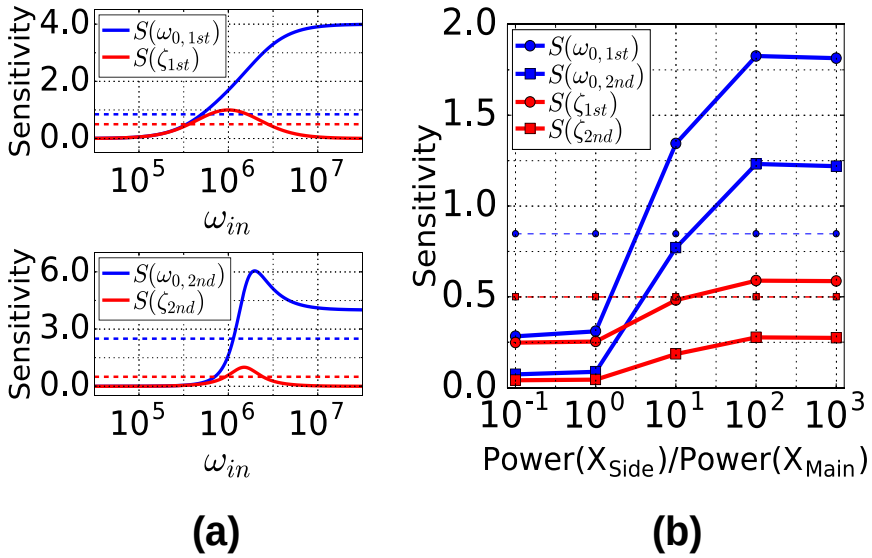


Figure 6.8. Sensitivity for the biquad low-pass filter example under (a) single-tone sinusoidal input with different frequency and (b) BPSK-modulated baseband input (X_{MAIN}) with different power of interferer input (X_{SIDE}).

6.4 Integrate and Fire Block

The model of an integrate-and-fire block [50] in Figure 6.9(a) exemplifies the slicer. The model parameter for the slicer is V_{TH}, which is changed from 0.6 to 0.9 during the experiments. The input for the block is given as a DC voltage to V_{IN}.

The example includes a simplified piecewise-linear model of PMOS transistor following the circuit primitives in XMODEL [33], with transconductance of 10 μA/V and threshold voltage of 0.4 V. Other model parameters are given as V_{DD}=1.2 V, D_{FB} = 10 pS, C_{INT} = 10 fF. The reset switch is assumed to be ideal, having infinite off-resistance and zero on-resistance.

The PMOS transistor operates as one of the three type of linear circuit elements following the operation region. First, it becomes a current source of $I_{DS} = 10(V_{SG} - 0.4)\mu A$ in its saturation region ($|V_{GS}| > 0.4$ and $|V_{DS}| > |V_{GS}| - 0.4$). Second, it becomes a resistor with R_{DS} = 100kOhm in its linear region ($|V_{GS}| > 0.4$ and $|V_{DS}| < |V_{GS}| - 0.4$). Finally, it becomes an opened terminal in its cut-off region ($|V_{GS}| < 0.4$).

Figure 6.9(b) shows the sensitivity of V_{TH} for simulations with applying various DC values on V_{IN}. For all the cases, the sensitivity is equal or larger than 0.5, indicating that the simulation scenario mostly covers the V_{TH}. This is because the node V_A is charged effectively by a constant current source or a constant resistor, following the assumed transistor model, where the charging waveform at V_A is always concave during increasing. The combination of a

low V_{IN} and large V_{TH} drives the PMOS transistor more in resistive region. When the transistor operates as the resistor, V_A increases as the step response of a 1-st order system and the slope of input waveform of slicer becomes smaller than the case when the transistor being current source. As a result, the sensitivity of slicer is higher for the larger V_{TH} and the lower V_{IN}.

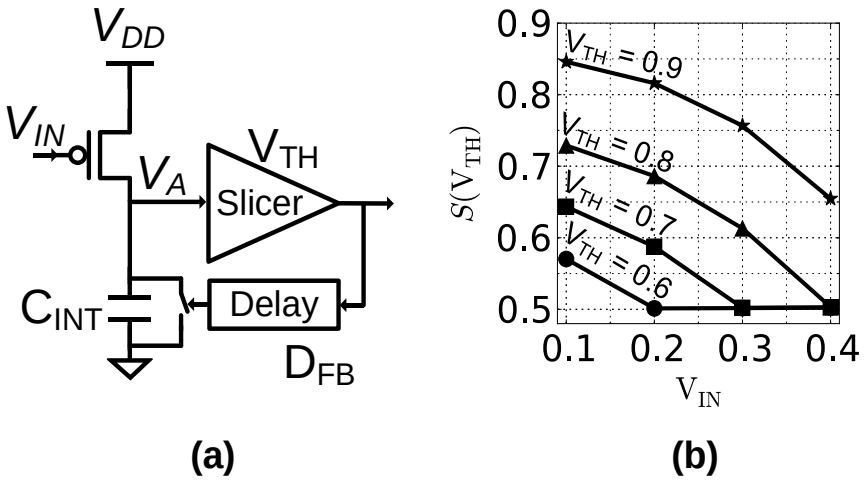


Figure 6.9. Block diagram of (a) integrate-and-fire cell and (b) sensitivity of the slicer's threshold under various values of DC input.

6.5 Delta-Sigma Modulator

Figure 6.10 shows the block diagram of a two-bit delta-sigma modulator that exemplifies the comparators. The parameters are $\{V_{TH1}, V_{TH2}, V_{TH3}\} = \{2, 0, -2\}$ and integrator gain = 10^6. The DAC in feedback path produces an output level of {–3, –1, 1, 3}, with respect to the two-bit input $D_{OUT[1:0]}$. The clock frequency for comparators are 1 GHz, which has sinusoidal timing jitter of frequency 100 MHz and amplitude 0.1 UI.

Figure 6.11 shows the sensitivity measured by simulations with applying a DC or sinusoidal input with various amplitudes, where the dashed lines represent coverage decision threshold for comparator derived in Chapter 5.7.

Figure 6.11(a) shows the measured sensitivity values for a DC input of amplitude V_{DC}, that ranges from –2.5 to 2.5. The sensitivity for each threshold value becomes sufficiently larger when the input level becomes near the threshold of each of the three comparators. The null for the sensitivity V_{TH3} at $V_{DC} = -2.0$ represents the limit-cycle behavior of entire feedback system, that the comparator's input only has samples far from V_{TH3}.

Figure 6.11(b) shows the result of applying a sinusoidal input of frequency 10 MHz and offset 0, with the amplitude V_{AMP} being swept from 0.1 to 2.4. For small V_{AMP}, the sensitivity for V_{TH2} is only high. On the other hand, V_{AMP} larger than 1.5 makes all V_{TH1-3} to have large enough sensitivity values to be covered.

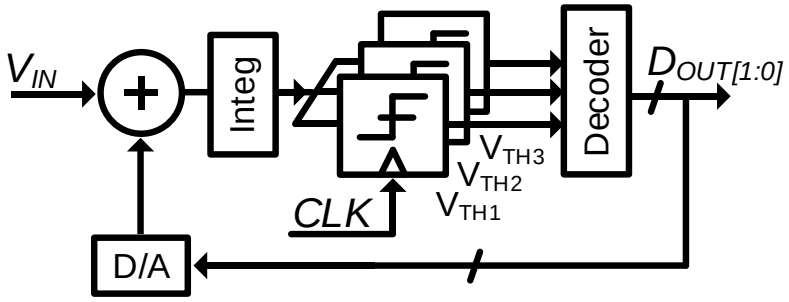


Figure 6.10. Block diagram of a two-bit delta-sigma modulator (DSM).

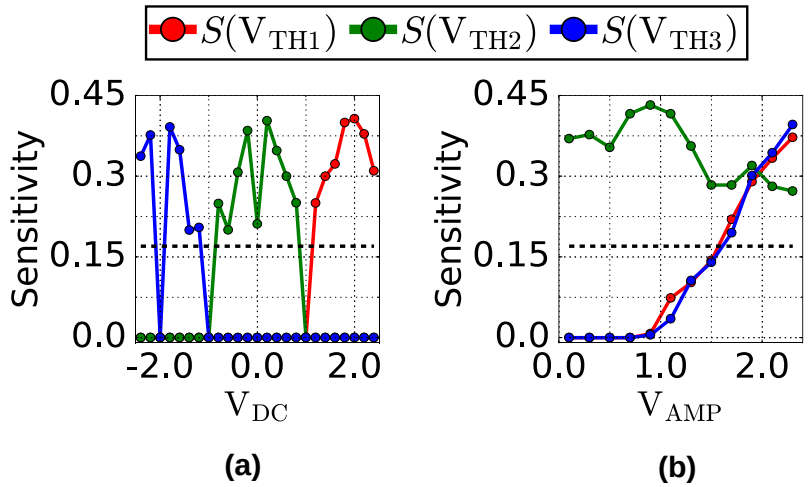


Figure 6.11. Sensitivity for the threshold of comparators by applying (a) DC input with various levels and (b) sinusoidal input with various amplitudes.

6.6 RF Receiver Analog Front-End

In this example, we applied the proposed coverage analysis to the simulation of an example of analog frontend in an RF receiver model [34]. Figure 6.12 shows the block diagram of a simplified example model of direct down-conversion receiver.

The sub-blocks of the model include a low-noise-amplifier (LNA), mixer, baseband low-pass filter (LPF), and programmable-gain amplifier (PGA). Specifically, the blocks include several DACs to control the gain or bandwidth of the primary signal path. Switches were used for the path configuration. A total of 108 model parameters were used in the example model. The topology of each blocks is basically same to that of Figure 3.2, except for the existence of switches for representing the bypass mode or other types of multi-modality.

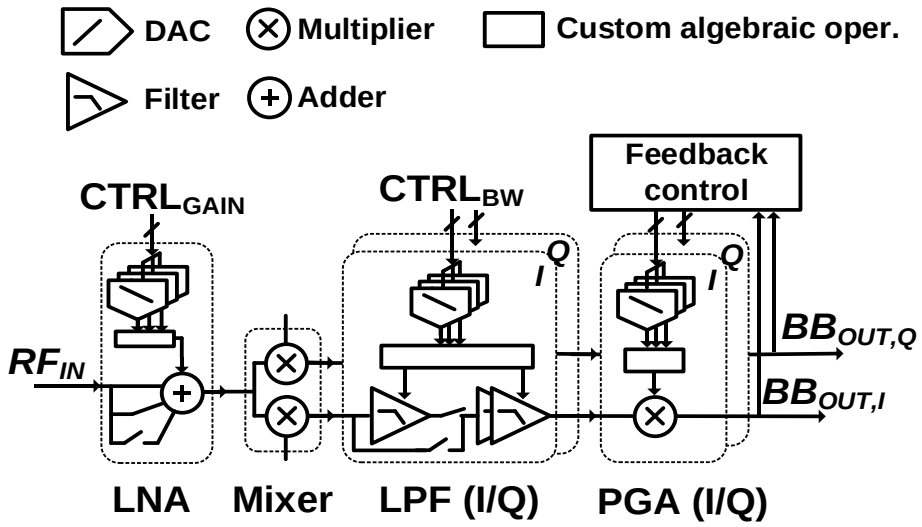


Figure 6.12. Block diagram of analog functional model in RF receiver frontend employing direct down-conversions scheme.

The simulation input consists of an analog RF-band input and digital codes for gain/bandwidth control, which are typical simulation items for verifying an analog frontend [35]. The first experiment is to repeatedly apply randomized input signals to the model and observe the increase of cumulative coverage. The randomization was on digital control signals and the signal amplitude for RF input, while the carrier frequency was fixed.

Figure 6.13 shows the cumulative coverage values as more simulation stimuli are added, for the three combinations of input generation scenarios listed in Table 6.1. For the three cases, the cumulative coverage starts near 40% and increases rapidly as more inputs are applied, converging to a different level with respect to the simulation type. Each simulation stimuli type eventually reaches ~45%, ~72%, ~90%, in which each covers the model parameters in the LNA, BBLPF, and PGA, respectively.

The remaining uncovered parameters are primarily single-to-differential connectors with the common-mode input as zero, for which the corresponding parameter values cannot be covered.

As an equivalence checking of two system models, we changed some parameters in the model and observe the difference under same simulation stimuli for the Figure 6.13. The difference is measured on final output ($BB_{OUT,I/Q}$) in Figure 6.12, using (4.1). We doubled the weights of a DAC primitive in each of the LNA, BBLPF, and PGA blocks, where the output of the DACs control the parameter of other blocks such as filter or scaler. The number of total weights for the selected DACs ranges from 3 to 6.

Figure 6.14 shows the histogram of the difference in final output for a single

run of simulation. The total simulation run lengths are 5 (1st row), 10 (2nd row), and 15 (3rd row). As the cumulative coverage increases (Figure 6.13), the number of samples for large output difference also increases.

Table 6.1. Conditions for generating input stimuli for RF receiver model example. The condition differs in randomizing condition for each block of LNA, BBLPF and the PGA.

Type #	Input stimuli generation method
1	Randomizing control code for LNA
2	Randomizing control code for BBLPF
3	Randomizing control code for PGA. (with enabling gain adaptation loop)

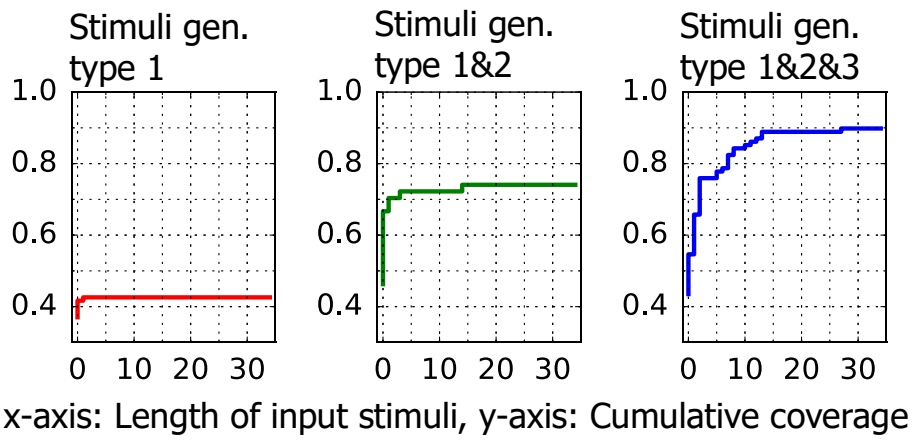


Figure 6.13. Parameter coverage value from three scenarios of simulation stimuli generation in table 3. The accumulative coverage increases and converges as more simulations are run with new input stimuli.

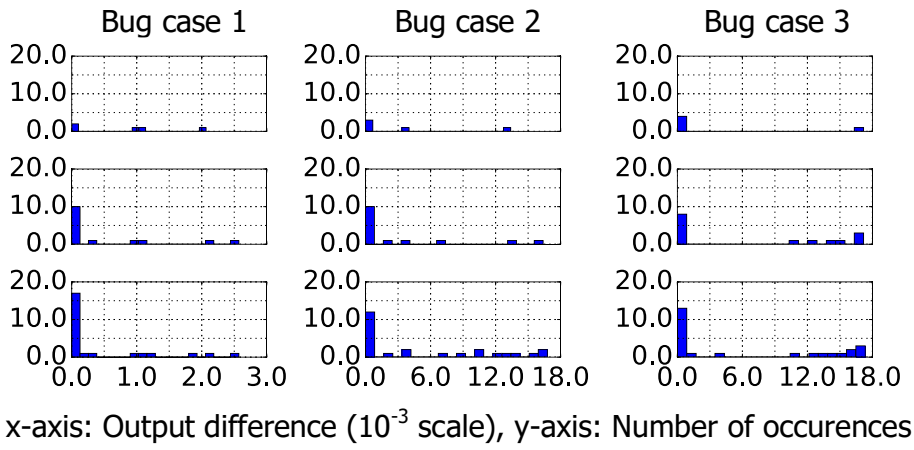


Figure 6.14. Histogram of the relative differences in final output for two models under comparison. The lengths of simulation stimuli are 5 (1st row), 10 (2nd row), and 15 (3rd row) for each figure.

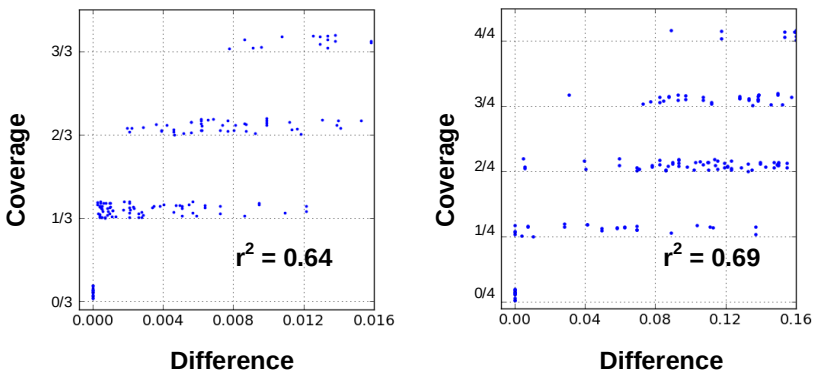


Figure 6.15. Visualization of the correlation in output difference due to the 2x parametric error in a DAC and the coverage for the block parameters under different random simulations.

6.7 Correlation to Fault Injection

We further investigated how the proposed coverage metric correlates with the likelihood of observing signal difference at the probe nodes, given the parameter change in a system model. Considering the symmetry of blocks, we selected total 9 independent DACs and doubled their parameters. The selected DACs are in LNA, PGA, or LPF block and have static parameters.

For each case of parameter change, simulations were performed 100 times with randomized stimuli. Then for each simulation run, we extract the parameter coverage on the bug-injected block and the relative difference in probe nodes. The probes nodes are the output of sub-block containing it.

From the samples of the values for coverage and difference, we extracted correlation coefficient (r^2) via linear regression. Table 4 lists the results, which ranges from 0.60 to 0.83, after further processing the coverage report with observability analysis. Figure 6.15 visualizes the correlation of samples from repetitive simulations.

Table 6.2. Correlation between the proposed coverage and actual signal difference in probe nodes under bug injection for each performance-controlling DACs in sub-blocks.

Bug-injected primitive blocks	Correlation coefficient
DACs in LNA	0.60, 0.69, 0.71
DACs in LPF	0.79, 0.80, 0.75, 0.73
DACs in PGA	0.83, 0.82

6.8 Future works

In this sub-chapter, we discuss on two future works of the coverage analysis. The first one is on the controllability analysis on internal blocks, which was partially addressed in previous Chapter 5.8. The second one is on automatic generation of simulation input stimuli for achieving high coverage. The controllability analysis and input generation are for understanding the uncovered parts of the design, after running simulations with sufficiently large number of randomly generated input stimuli.

The purpose of controllability analysis is to know that whether the uncovered parts will be eventually covered by generating more input stimuli. When the structure of DUV and the condition for random input generation results in a truly uncoverable part in design, a verification engineer should judge and waiver on that part. One example is that a design has redundant part, which is independent to the operation of system.

However, proving that a part of a design can never be covered is not trivial as it is equivalent to show that the all of input signal for an internal block has low sensitivity than the decision threshold. In this sense, the input generation problem is formulated as to search an external input signal that maximizes the sensitivity for an internal block.

The following sub-sections briefly discusses on numerical techniques applicable for the two problems, using satisfiability-modulo theory (SMT) solver and optimization based on response surface method (RSM).

6.8.1 Controllability Analysis Using SMT

This sub-chapter discusses on the usage of SMT solvers with an example shown in Figure 6.16. The example model represents parallel placement of amplifiers, which can be observed in multi-modal analog functional blocks [3]. The model consists of four scalers and switches. The outputs of four amplifiers $\{ y_k(t) \}$'s are finally mapped to two signals $\{ w_1(t), w_2(t) \}$ through adders that represent signal combiners. The gain of amplifiers is assumed to be controllable and the four amplifiers receives a common input $x(t)$. Also, four logic signals (sw_1-sw_4) exist for controlling the switch at the output of each amplifiers. If a switch is open, the value of y is set to zero value.

The problem is to show that whether the weight parameter corresponding for the adder taking z_2 and z_3 as input is coverable or not, under the condition for all possible input values for $x(t)$ and the configuration for switches.

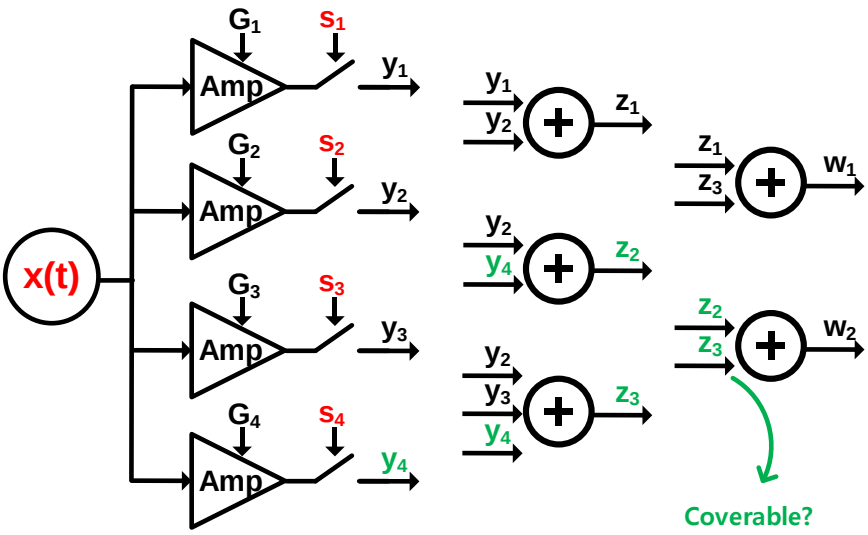


Figure 6.16. A model including switched parallel blocks and the combination of outputs. The controllability

An SMT solver [40] takes logical and part of arithmetic constraints on variables as input and proves whether the constraints are satisfiable or not. If the constraints are satisfiable, the example of variables are also provided. Specifically, SMT solvers can take real variables in representing analog signals.

We can use the SMT solver to prove that a parameter is coverable in given condition of simulation and model structure. To do so, we apply three types of constraints, that are the simplified condition for the analog input $x(t)$, the structure and functionality of model blocks, the condition for covering the parameter.

Figure 6.17 shows an example source code using Z3 solver in python, to show that the weight parameter for the adder's input taking the signal z_2 is coverable or not. As a demonstration example, the input $x(t)$ is assumed to a DC signal ranges from 0 to 1, Also, constraints exist on the condition for the logic values for controlling switch states.

```
1:  From z3 import *
2:  S = Solver()
3:  s1 = Bool('s1'); x = Real('x'); y1 = Real('y1') …w2 = Real('w2');
4:  S.add(x > 0); S.add(x < 1)
5:  S.add(g1 > 5); S.add( g1 < 10); … S.add( g4 < 10);
6:  S.add(Or(And(y1==g1*x, s1), And( y1==0, Not(s1))));
    …
7:  S.add(Or(And(s1 == s2), And(s3 == s4)))
8:  S.add(z1 == y1+y2); S.add(z2 == y1 +y3) ;
    …
9:  S.add(z2*z2 > w2*w2*0.5)
10: Print S.check()
```

Figure 6.17. The part of source code for proving the coverability of a parameter with Z3 solver in python.

The line 3 of the example code states the name of signals in the example model. Then, the constraints on input is stated as in line 4. Also, the possible ranges for the gain of the four amplifiers are stated in line 5. The function of output-enable switches in each blocks, and the Boolean constraints on the switch control signals are stated through line 6-7 and the information of signal connection at the adders are stated in line 8. Finally, the condition for covering the target weight parameter is stated as line 9, following the Chapter 4.1.1 and 5.2.

With all the constraints and model information, line 10 executes the SMT solver and the satisfiability is computed. If the result is *sat,* the example of signal values, including the switch configurations and input $x(t)$ is given. Otherwise, the result of *unsat* tells that covering the target parameter is impossible in the model and simulation condition.

However, in using SMT, the possible form of expressions for the solver can have limitations in representing various analog waveforms. Specifically, having the result of *unsat* matters, as it justifies putting waiver on uncovered part of design after long simulations. For that cases, conservative approximation techniques should be investigated so that the result of *unsat* from the SMT solver can sufficiently imply that the same result for real situation of simulations.

6.8.2 Automatic Input Generation

In discussing the automatic input generation to enhance coverage, the time spent for generating the input matters. The input generation problem is to find an input that can enhance the sensitivity values for the uncovered parameters, which involves predicting the sensitivity values for each parameter and to select an input that maximizes the number of newly covered parameter by running additional simulation.

Figure 6.18 shows to use response surface modeling (RSM) techniques and optimization algorithms for generating input signals. Given the existing simulation results of DUV, the sensitivities values for each uncovered parameter in DUV can be approximated by the response models. Then, the optimization engine finds promising input signals under the constraint of valid inputs. In searching the promising inputs, the RSMs are used as the predictive model for the sensitivity values achieved by applying an input signal.

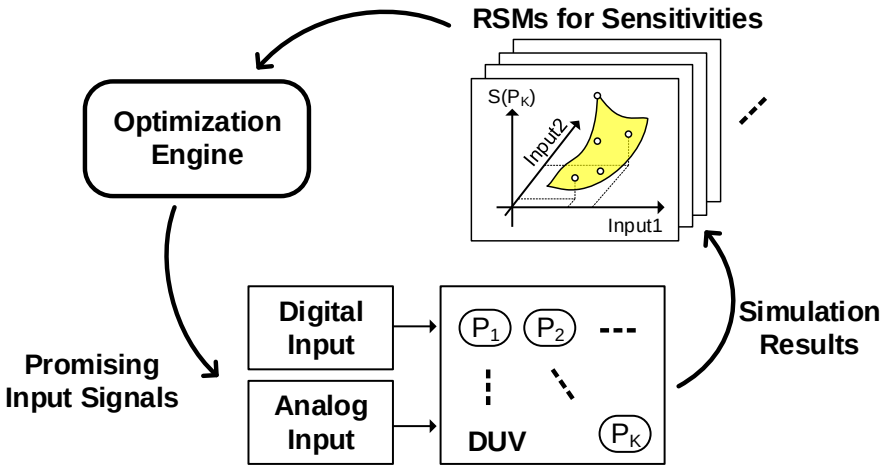


Figure 6.18. The optimization-based approach for generating promising input stimuli for enhancing the parameter coverage on DUV.

However, the time spent for generating input increases as the number of uncovered parameter increase. As the constrained random generation of input typically gives rapid coverage enhancement, the optimization-based approach should be used after the coverage enhancement from random generation becomes marginal. Otherwise, the input generated from the optimization algorithms enhances the less amount of coverage, while consuming more computation times to predict sensitivity values of large number of uncovered parameters.

The automatic generation would be beneficial when the sample of initial simulation traces and sensitivity values are sufficiently accumulated, and when the remaining parameters are hard to cover by random generation approaches, justifying the time spent for input generation.

Another concern in using optimization algorithms is to construct the searching space. As most of analog functional models are simulated together with digital blocks, which receives digital inputs, the searching space of input stimuli includes Boolean variables. The digital inputs should be distinguished as the ones for block enable functions and the other ones for controlling analog parameters. For the latter cases, a multiple digital control signals resulting in a single type of analog parameter (e.g., the bandwidth for a specific instance of a block) can be merged to reduce the number of variables. However, reducing the signals controlling block enable/disable can be more challenging, as the RSMs resulting from each configuration of switches will exhibit less correlation.

Chapter 7

Conclusion

The ever-increasing complexity of analog blocks in IC-level operation poses new challenges for the verification, which calls for the coverage analysis for the analog functional models in system level simulations. Together with the already established coverage analysis on digital HDL models, the coverage analysis on analog functional model parameters can make the system-level verification of mixed-signal ICs more complete.

This dissertation presented a theoretical basis to quantify the coverage of functional simulations with analog/mixed-signal models. Specifically, the sensitivity functions and decision thresholds for different types of analog functions are derived, that are applicable for various instances of analog functional blocks. The coverage can indicate whether a given simulation can verify the correctness of a given model parameter value, whose usefulness is validated with the typical simulations of several block-level examples and a system-level simulation example of analog and mixed signal circuits.

While the work is based on the premise that analog/mixed-signal system models are described as a collection of basic primitives, it is a meaningful step towards a complete coverage analysis on analog/mixed-signal simulations.

Bibliography

[1] S. Balasubramanian and P. Hardee, “Solutions for Mixed-Signal SoC Verification Using Real Number Models,” Cadence Design Systems, 2013.

[2] K. Kishore, et al., “Solutions for mixed-signal SoC verification,” Cadence Design Systems, 2009.

[3] J. Lee, et al., “21.6 A Sub-6GHz 5G New Radio RF Transceiver Supporting EN-DC with 3.15Gb/s DL and 1.27Gb/s UL in 14nm FinFET CMOS,” *in IEEE Int. Solid State Circuit Conf. Dig. Tech. Papers*, pp. 354-356, 2019.

[4] M. Jeong, et al., "Multi-band multi-mode wireless connectivity SoC for 802.11 a/b/g/n, BT 4.0 and NFC," ISOCC, pp. 163-164, 2014.

[5] Cadence, “16Gbps Multi-Link and Multi-Protocol PHY”, Cadence Design Systems, https://ip.cadence.com/uploads/1207/cdn-dsd-int-ser-10-gbps-multi-link-multiprotocol-phy-ip-pdf. accessed July 15, 2019.

[6] M. Konijnenburg et al., “28.4 A battery-powered efficient multi-sensor acquisition system with simultaneous ECG, BIO-Z, GSR, and PPG,” *in IEEE Int. Solid State Circuit Conf. Dig. Tech. Papers*, pp. 480-481, 2016.

[7] H. Chang and K. Kundert, “Verification of Complex Analog and RF IC Designs,” *Proc. IEEE*, pp. 622-639, Mar. 2007.

[8] G. Nunn, et al. “Using Digital Verification Techniques on Mixed-Signal SoCs with CustomSim and VCS,” White Paper-Synopsys, Mar. 2011.

[9] J. Bergeron, “Writing testbenches using SystemVerilog,” Springer Science & Business Media, 2007.

[10] T. R. Halfhill, “The Truth Behind the Pentium Bug: An error in a lookup table created the infamous bug in Intel's latest processor," *BYTE,* Mar. 1995.

[11] Y. Sha, et al., “On Code Coverage Measurement for Verilog-A,” *in Proc. IEEE International High-Level Design Validation and Test Workshop*, pp. 115-120, Nov. 2004.

[12] A. Fürtig, et al., “Comparing code coverage metrics for analog behavioral models,” *in Proc. Int. Conf. Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*, pp. 1-4, 2017.

[13] A. Ain, et al., “Feature Based Coverage Analysis of AMS Circuits,” *in proc. IEEE Computer Society Annual Symp. on VLSI*, pp. 423-428, 2018.

[14] A. Furtig, et al., “Feature Based State Space Coverage of Analog Circuits,”

in Proc. Forum on Specification and Design Languages, pp. 1-7, Sep. 2016.
[15] A. Fürtig, et al., "Novel Metrics for Analog Mixed-Signal Coverage," *in Proc. IEEE Int. Symp. Design and Diagnostics of Electronic Circuits & Systems*, pp. 97-102, Apr. 2017.
[16] T. Dang and T. Nahhal, "Coverage-Guided Test Generation for Continuous and Hybrid Systems," *Formal Methods in System Design*, pp. 183-213, Apr. 2009.
[17] S. Steinhorst and L. Hedrich, "Improving Verification Coverage of Analog Circuit Blocks by State Space-Guided Transient Simulation," *in Proc. IEEE Int. Symp. Circuits and Systems*, pp. 645-648, May 2010.
[18] N. Khan, et al., "Metric Driven Verification of Mixed-Signal Designs*," in Proc. Design and Verification Conf.*, Mar. 2011.
[19] N. Khan and Y. Kashai, "From Spec to Verification Closure: a case study of applying UVM-MS for first pass success to a complex Mixed-Signal SoC design," *in Proc. Design and Verification Conf.*, Mar. 2012.
[20] C. Liang, "Mixed-signal verification methods for multi-power mixed-signal System-on-Chip (SoC) design," *in Proc. IEEE Int' Conf. ASIC*, pp. 1-4, 2013.
[21] S. Sebastian, et al. "Coverage-Driven Mixed-Signal Verification of Smart Power ICs in a UVM Environment," *in Proc. IEEE European Test Symp.*, pp. 1-6, May 2017.
[22] C. Liang, et al., "UVM-AMS based sub-system verification of wireless power receiver SoC," *in Proc. IEEE Int. Conf. Solid-State and Integrated Circuit Technology*, pp. 1-3, Oct. 2014.
[23] F. Yang, et al., "Metric Driven Mixed-Signal Verification Methodology and Practices for Complex Mixed-signal ASSPs", [Online] Available: http://events.dvcon.org/2014/proceedings/papers/01P_23.pdf. *Design and Verification Conf.*, Mar. 2014.
[24] B. C. Lim, et al., "An Efficient Test Vector Generation for Checking Analog/Mixed-Signal Functional Models," *in Proc. ACM/IEEE Design Automation Conf.*, pp. 767-772, Jun. 2010.
[25] M. Horowitz, et al., "Fortifying Analog Models with Equivalence Checking and Coverage Analysis," *in Proc. ACM/IEEE Design Automation Conf.*, pp. 425-430, Jun. 2010.
[26] A. Balivada, et al., "Verification of Transient Response of Linear Analog Circuits," *in Proc. IEEE VLSI Test Symp.*, pp. 42-47, Apr. 1995.

[27] A. Ain, et al., “Feature Indented Assertions for Analog and Mixed-Signal Validation,” *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, pp. 1928-1941, Nov. 2016.

[28] N. B. Hamida and B. Kaminska, “Analog circuit testing based on sensitivity computation and new circuit modeling,” *in Proc. IEEE Int. Test Conf.*, pp.652-661, Oct. 1993.

[29] A. Sangiovanni-Vincentelli, et al., “Benefits and Challenges for Platform Based Design,” *in Proc. ACM/IEEE Design Automation Conf.*, pp. 409-414, Jun. 2002.

[30] B. C. Lim and M. Horowitz, “An Analog Model Template Library: Simplifying Chip-Level, Mixed-Signal Design Verification,” *IEEE Trans. Very Large Scale Integr. Syst.,* pp. 193-204, Jan. 2019.

[31] H. D. Foster, "Trends in Functional Verification: A 2014 Industry Study," *in Proc. ACM/IEEE Design Automation Conf.*, pp. 1-6, Jun. 2015.

[32] A. Singh and P. Li, “On Behavioral Model Equivalence Checking for Large Analog/Mixed Signal Systems,” *in Proc. Int. Conf. Comput. Aided Design*, pp. 55-61, Nov. 2010.

[33] Scientific Analog (2019), XMODEL™, www.scianalog.com.

[34] P.-I. M, et al., “Analog-Baseband Architectures and Circuits for Multistandard and Low-Voltage Wireless Transceivers,” Springer Science & Business Media, 2007.

[35] S. Joeres and S. Heinen, “Functional Verification of Radio Frequency SoCs using Mixed-Mode and Mixed-Domain Simulations,” *in Proc. IEEE Beh. Modeling Simul. Workshop*, pp. 144-149, Sep. 2006.

[36] F. Fallah, et al., “OCCOM-Efficient Computation of Observability-Based Code Coverage Metrics for Functional Verification,” *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, pp. 1003-1015, Aug. 2001.

[37] J. Kim, et al., “Variable domain transformation for linear PAC analysis of mixed-signal systems,” *in Proc. IEEE/ACM Int. Conf., Computer-Aided Design*, pp. 887-894, 2007.

[38] J. Kim, et al., “Leveraging designer's intent: A path toward simpler analog CAD tools,” *in Proc. IEEE Custom Integrated Circuits Conference*, pp. 613-620, 2009.

[39] B. C. Lim, et al., “Digital Analog Design: Enabling Mixed-Signal System Validation,” *IEEE Design & Test*, vol. 32, no. 1, pp. 44-52, Feb. 2015.

[40] D. Yurichev, “SAT/SMT by Example”, accessed July 19, 2019,

https://yurichev.com/writings/SAT_SMT_by_example.pdf.
[41] J. Chen, et al. “Mixed-signal methodology guide”, Lulu. com, 2012.
[42] C. Visweswariah, et al., “Model development and verification for high level analog blocks,” *in Proc. ACM/IEEE Design Automation Conf.*, pp. 376-382, Jun. 1988.
[43] A. Devgan, “Accelerated design of analog, mixed-signal circuits with fineSim™ and titan™,” *in Proc, Int. Conf. SoC Design*, pp. 282-286, 2009.
[44] F. Speicher et al., “Methodology for improved event-driven system-level simulation of an RF transceiver subsystem for wireless SoCs,” *in Proc. Int. Conf. Design & Technology of Integrated Systems in Nanoscale Era*, pp. 1-4, 2018.
[45] K. Kundert and H. Chang, “Verifying all of an SOC-analog circuitry included,” *IEEE Solid-State Circuits Magazine*, vol. 1, no. 4, pp. 26-32, 2009.
[46] K. D. Jones, et al., “Some “real world” problems in the analog and mixed signal domains,” *in Proc. Designing Correct Circuits*, Apr. 2008.
[47] W. Hartong, et al., “Model checking algorithms for analog verification,” *in Proc. ACM/IEEE Design Automation Conf.*, pp. 542-547, Jun. 2002.
[48] J. F. Bulzacchelli, “Equalization for Electrical Links: Current Design Techniques and Future Directions,” *IEEE Solid-State Circuits Magazine*, vol. 7, no. 4, pp. 23-31, 2015.
[49] J. Peng, et al., “SQNR Improvement Enabled by Nonuniform DAC Output Levels for IM-DD OFDM Systems,” in IEEE Photonics Journal, vol. 9, no. 2, pp. 1-11, 2017.
[50] G. Indiveri, “A low-power adaptive integrate-and-fire neuron circuit,” *in Proc. Int. Symp. Circuits and Systems*, May 2003.

초 록

본 연구는 아날로그 기능 모델의 모의 시뮬레이션에 대한 커버리지를 정량화 하는 한 방법을 제시한다. 제안하는 방법은, 수행한 시뮬레이션이 아날로그 블록 모델의 파라미터 (전압 이득, 대역폭, 오프셋) 값이 올바른 지 확인하는데 유용한지를 평가할 수 있다. 제안하는 방법은 어떠한 파라미터 값이 변화 시 시뮬레이션 결과 파형이 얼마나 변화할 지를 민감도 값으로 산출한 뒤, 그 민감도가 어떤 임계 값보다 클 경우 해당 파라미터가 잘 확인되었다고 규정한다.

본 논문이 주요하게 제시하는 부분은 두 가지이다. 첫 번째는 아날로그 블록이 받을 수 있는 다양한 범위와 형태의 입력 신호 파형에 대하여 적용 가능한 민감도 측정 함수를 제시한 것이다. 두 번째는, 요소 블록에 대한 확률적 입력 신호 공간을 정의하고, 이로부터 정보 이론에 기반한 최적의 임계 값을 결정하는 원리를 제시한 것이다. 본 연구에서는 제안하는 방법을 다양한 아날로그 모델에 적용하기 위해, 덧셈기, 곱셈기, 필터, 디지털-아날로그 변환기, 아날로그-디지털 변환기 등의 대표적인 요소 블록들을 표준화하고 각각에 대한 분석을 수행하였다.

제안한 방법은 등화기, 고차 필터, 재입력 변조기 등 유무선 통신에 흔히 쓰이는 아날로그 블록들에 대한 통상적 시뮬레이션의 커버리지를 측정하여, 각 시뮬레이션 입력에 대하여 실제로 검증되는 파라미터가 무엇인지를 식별하였다. 또한 무선통신 수신기 내의 아날로그 전단부 동작 모델에 대한 실험 결과, 시뮬레이션

입력 생성 조건에 따라 모델 전체에 대한 커버리지가 40%에서 90%까지 변화함을 밝혔다. 또한, 제안하는 방법에 의해 측정된 커버리지 값과 실제 파라미터 값이 변화 시, 시뮬레이션이 이를 확인하는 정도 간의 상관관계를 분석하여, 0.60 ~0.83 정도의 상관계수를 가짐을 확인하였다.

주요어 : 시뮬레이션 기반 검증, 파라미터 커버리지 분석, 파형 민감도 분석, 확률적 입력 공간

학 번 : 2013-20860