Partial Simulation-Driven ATPG for Detection and Diagnosis of Faults in Analog Circuits

Sudip Chakrabarti and Abhijit Chatterjee
School of Electrical & Computer Engineering, Georgia Institute of Technology, Atlanta, GA.

Abstract

In this paper, we propose a novel fault-oriented test generation methodology for detection and isolation of faults in analog circuits. Given the description of the circuit-under-test, the proposed test generator computes the optimal transient test stimuli in order to detect and isolate a given set of faults. It also computes the optimal set of test nodes to probe at, and the time instants to make measurements. The test generation program accommodates the effects introduced by component tolerances and measurement inaccuracy, and can be tailored to fit the signal generation capabilities of a hardware tester. Experimental results show that the proposed technique can be applied to generate transient tests for both linear and non-linear analog circuits of moderate complexity in reasonably less CPU time. This will significantly impact the test development costs for an analog circuit and will decrease the time-to-market of a product. Finally, the short duration and the easy-to-apply feature of the test stimuli will lead to significant reduction in production test costs.

1. Introduction

In recent years, analog circuit testing has received a lot of attention due to the large number of analog and mixed-signal applications in various fields. Traditional test methods for analog circuits rely on *specification testing*, in which some or all response parameters are checked for conformity to the design specifications. However specification testing is time consuming and expensive. As an alternative, fault-based test strategies are being increasingly used. Fault-based techniques test for the *presence* (or *absence*) of physical manufacturing-related defects (*faults*), thereby providing a quantitative estimate of the effectiveness and completeness of the testing process. They also allow the test procedure to test only for the most likely group of faults induced by a manufacturing process.

In fault-driven test strategies, appropriate parameters of the test waveform (DC, AC or transient) are determined, which can excite the faults and make them observable in the measurement space. DC test generation schemes have been proposed in [1] and [2] that use high-level iteration and linear approximation of the circuit to generate DC test stimuli. However, DC tests are not adequate for detecting parametric faults. In [3] and [4], sensitivity-based algorithms have been proposed to generate frequency-domain (AC) tests. AC testing requires the *circuit under test* (CUT) to settle down to a steady state before measurements can be made, and this leads to longer test times. In transient testing, an aperiodic waveform is applied to the CUT and its response is measured during application of the transient. By exciting the CUT with pulses and ramps whose frequency spectrum stretches over a wide range of frequencies, we can make all faults visible in the measurement space. Transient test generation algorithms using quadratic programming [5], *minimax* optimization [6] and bilinear transformation [7] have been proposed in recent past. In [8], a diagnostic transient test generator has been proposed that uses optimization techniques at every time point to determine the transient test stimuli for both fault detection and fault isolation. However, all of the analog test generation techniques proposed so far suffer from the following limitations. Firstly, the quality of tests is highly influenced by the presence of local extrema of the objective function. Also, the *test nodes* (circuit nodes at which measurements can be made) and the *sampling time instants* (time at which measurements are made) are given as input by the user. In practice, the user may not have enough information about the circuit to pre-specify the *best* test nodes or sampling time instants. This leads to inferior test stimuli (low fault coverage) and / or longer test times.

In this paper, we propose a novel transient test generation methodology for both fault detection (to determine whether the circuit is *good* or *bad*) and fault isolation (to identify the *faulty* sub-circuit). The input to the proposed test generator consists of the circuit description, failure modes of various circuit components and the tester specifications. The tester specifications provide information (maximum slew-rate, peak-to-peak voltage, etc.) about the kind of waveforms that can be generated and the type of measurements that can be made on a hardware tester. The output of the test generator consists of test stimuli, test nodes and sampling time instants that are *optimally* sufficient to detect (or isolate) all the failure modes (faults) in the *circuit under test* (CUT). The proposed method is appli-

Research supported by AFRL under contract E21E77

cable to both linear and non-linear circuits and systems of moderate complexity, and accommodates component tolerance effects and measurement inaccuracies. A complete test generation system for both fault detection and fault isolation has been implemented and tested on several circuits.

The paper is organized as following: Section 2 introduces the basic concepts in analog test generation and the fault models used. Section 3 presents an overview of the test generator. The key test generation and optimization module is described in Section 4, followed by a brief discussion on the fault simulation strategy in Section 5. The evaluation of the objective function is described in Section 6, followed by some experimental results in Section 7.

2. Basic concepts and approach

In this section, we introduce some key concepts that are used in the rest of this paper. First, we present the concept of automated test generation and its application to analog circuits followed by a brief description of the fault models used in our approach.

2.1 Test generation for analog circuits

Consider the leapfrog filter shown in Figure 1. The filter consists of six sub-circuits called *Part 0*, *Part 1*, ..., *Part 5* as shown in the figure. Each sub-circuit, in turn, consists of resistors, capacitors and an op-amp. The input to the filter is applied at circuit node *in* and the output is observed at circuit node *out*. Now, let us assume that we are given a faulty instance of the leapfrog filter in which the resistance of resistor R5 is 75% of its nominal value (10 kΩ). Let us further assume that our goal is to do pass / fail testing, i.e. to differentiate this faulty circuit from its fault-free instance. In order to do that, we are allowed to apply a transient test stimulus at input node *in* and we can make node voltage measurements at any circuit node at any time instant. Note that, the parameters of the transient test stimulus, i.e., slew-rate, frequency, etc., are governed by the signal generation capabilities of the testing equipment. Moreover, in most practical analog and mixed-signal circuits only a few of the circuit nodes can be probed at in order to make a test measurement. Therefore, the testing process must use minimum number of test nodes in order to detect a given list of faults. Finally, the time spent on a real

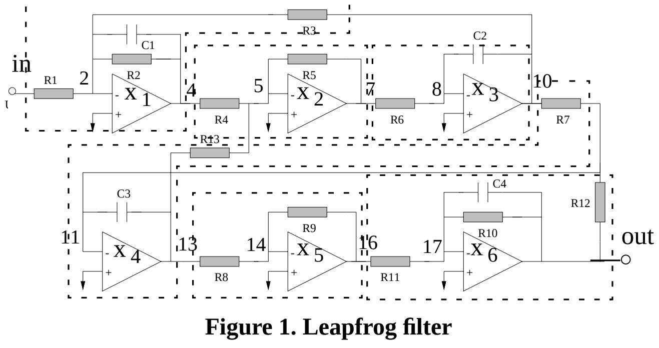


Figure 1. Leapfrog filter

tester is extremely expensive. Therefore, the interval from the time at which the test stimulus is applied to the time at which the measurement is made must be as small as possible. We call this time interval *sampling delay.*

Based on the above discussion, the requirements of an analog pass / fail test generator can be summarized as the following. Given the circuit description, the tester specifications and the list of targeted faults, the objective of the test generator is to determine the minimal set of test stimuli that can detect the maximum number of faults (i.e., maximize *fault coverage*), using least number of test nodes (i.e., minimize *test access*) and minimum sampling delay. Additionally, in case of test generation for fault isolation, the test stimuli must also be able to differentiate among faults from various sub-circuits. Note that, a fault can be differentiated from another fault (or from the fault-free instance) only when the corresponding measurement in the former differs from the latter by a certain minimum *threshold*. This threshold depends on component tolerances and measurement inaccuracy and is computed by the techniques described in [9]. In this paper, we assume the threshold is given as an input to the test generation program.

Let us consider the waveforms shown in Figure 2. Figure 2(a) shows the test stimulus that detects the fault in R5 (described earlier) within the leapfrog filter circuit. The corresponding waveforms at five different test nodes within the filter circuit are shown in Figure 2(b) through Figure 2(f). The threshold for fault detection is 0.5 V, and node voltage measurements at test nodes are used for simplicity (although any other transient measurement supported by the testing equipment could be used). From Figure 2, we can see that the fault at R5 is detected at test node 7 with minimum sampling delay.

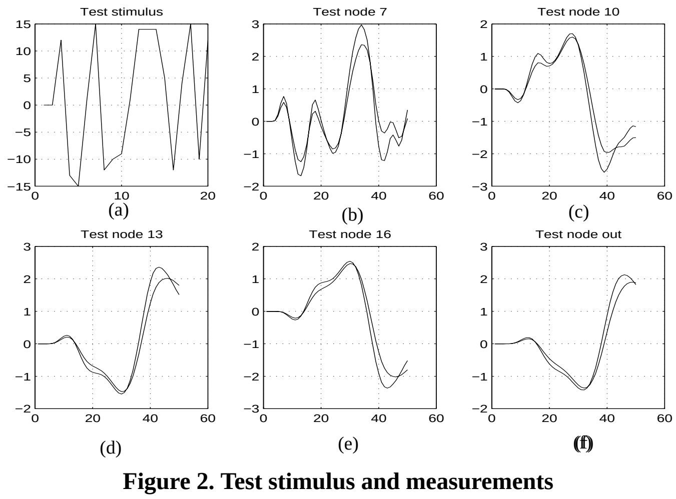


Figure 2. Test stimulus and measurements

2.2 Fault models

We assume that for each circuit component, a list of *catastrophic* (shorts and opens) and *parametric* (deviations outside tolerance bands) faults is given. It is assumed that the *fault-list* (list of all possible faults) is generated by *inductive fault analysis*

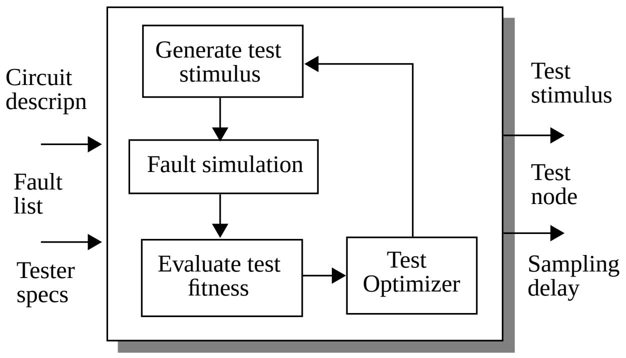


Figure 3. Proposed test generation methodology

tools [10] or from manufacturing defect data. Faults in interconnections between circuit components are subsumed into the fault-list of the corresponding components and the existence of a *single faulty component* is assumed.

3. Overview of the proposed approach

Consider Figure 3, which shows an overview of the proposed test generation methodology. The key components of the proposed test generator are the following:

- **Test stimulus generation and optimization:** This module uses a genetic optimization technique to compute the transient test stimuli that maximize fault coverage with minimum test access and sampling delay.
- **Fault simulation:** For a given test stimulus, the fault simulation module computes the response of the circuit in presence of each fault in the fault list. This step is the most time consuming, and therefore, a heuristic fault simulation strategy has been proposed to minimize the test generation time.
- **Evaluation of test fitness:** Given a test stimulus and the corresponding measurements in presence of all faults in the fault list, this module determines the *fitness* (i.e., objective function) of the test stimulus with respect to other test stimuli in the search space.

The input to the test generator consists of the following information: (a) <u>*circuit description*</u>: SpectreHDL behavioral models of the circuit components and their connectivity information in Spectre netlist format, (b) <u>*fault list*</u>: list of faults targeted for test generation, and (c) <u>*tester specifications*</u>: information on the type of test stimuli that can be generated or the measurement that can be made on the hardware tester, etc.

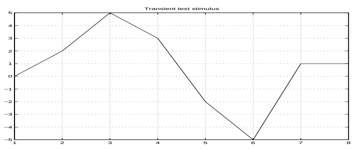
Given the above input, the test generation software generates the following output for each fault in the fault list: (a) <u>*test stimulus*</u>: the transient waveform (to be applied at the input of the CUT) that detects (or isolates) the corresponding fault, (b) <u>*test node*</u>: the circuit node at which the specified measurement is to be made, and (c) <u>*sampling delay*</u>: the time instant at which it is to be made.

In the following sections, we describe each component of the proposed test generator in more details.

4. Test stimulus generation and optimization

The goal of our approach, as described earlier, is to compute the set of test stimuli that maximize the fault coverage while minimizing test access and sampling delay. Therefore, the problem of test stimulus generation is an optimization problem in principle. Here, we have used a genetic optimization technique in our attempt to find the *globally optimal* test stimulus. Each transient test stimulus, <u>*T*</u>, is coded using a genetic string (an *n*-tuple), [t_1, t_2, ..., t_n], as shown in Figure 4. The number (n) of *alleles* in each string depends on the maximum slew-rate specification (of the signal generator) and is equal to the number of corner points in the piece-wise linear transient stimulus. The allowed range of voltage values is determined by the power supply levels of the CUT. The test stimuli are generated using the following rules of genetic selection, cross-over and mutation:

- <u>*Selection*</u>: The selection of strings is biased towards strings with higher *fitness* so that the average fitness of successive populations tends to increase. To select the parent strings, we have used *tournament selection*, in which two strings are picked and the better one is selected (with certain probability) for reproduction.
- <u>*Cross-over*</u>: During cross-over, genes from each parent string is combined to create child strings. We have chosen *uniform cross-over* in which each gene of the parent strings is chosen with a certain uniform probability and the parent strings are then crossed over at the selected gene to yield two child strings.
- <u>*Mutation*</u>: Once created, the genes of the child strings are selected with a certain probability and replaced with another selected at random. In our case, the gene undergoing mutation is replaced with a voltage value selected at random within the allowed range of voltage values.



0 2 5 3 -2 -5 1 1

Genetic string

Figure 4. Encoding of transient test stimulus

The generation and optimization of test stimuli can be described as following. The algorithm starts with a random population of test stimuli. Next, for each test stimuli, every fault in the given fault list is simulated and measurements are made at each test node. Based on the measurements, the fitness of each test stimulus is computed (Section 6) and the best stimuli are then chosen for reproduction. The test stimuli with inferior fitness are replaced with the newly generated child stimuli and the above process is repeated. The goal of test optimization is to move towards a better population with each successive iteration and the process continues till all the faults in the fault list are detected (or isolated). Also, pre-specified limits on the test development time can provide an alternate exit strategy.

5. Fault simulation

Given a test stimulus, fault simulation determines the measurement values in presence of each fault in the fault list. This involves numerous circuit simulations and hence contributes to a significant portion of the test development time. In this paper, we propose a heuristic fault simulation strategy in order to reduce the complexity of fault simulation. Consider an analog / mixed-signal circuit with M sub-circuits, P_1, P_2, ..., P_M. Assume that we have K candidate test stimuli, T_1 ,..., T_K, generated during test optimization (Section 4), and our goal is to compute the corresponding fault coverage, test access and the minimum sampling in the shortest possible time. At this point, let us introduce some key notations that will help explain the proposed fault simulation approach. Given a test stimulus T_j, we can identify a fault in each sub-circuit P_i that is *hardest to detect*, i.e., the measurements in presence of this fault are the *closest* to those in the fault-free case. For test stimulus T_j, the *h*ardest to detect faults in the M sub-circuits are denoted by an M-dimensional vector $\underline{f}(T_j)$. We also introduce a heuristic function $h(T_j)$ which is used as a similarity measure between two candidate test stimuli. We now propose that if $h(T_i) \leq h(T_j) \leq h(T_k)$, and if $\underline{f}(T_i) = \underline{f}(T_k)$ then $\underline{f}(T_i) = \underline{f}(T_j) = \underline{f}(T_k)$ with a certain probability. Therefore, if we compute $\underline{f}(T_i)$ and $\underline{f}(T_k)$ using partial fault simulation and fault ordering techniques as described in [11], then we can predict the hardest to fault vector $\underline{f}(T_j)$ for test stimulus T_j without any additional fault simulation. However, in case $\underline{f}(T_i) \neq \underline{f}(T_k)$ we need to actually determine $\underline{f}(T_j)$ via partial fault simulation. The heuristic function $h(T_j)$ is computed from the response of the fault-free circuit to test stimulus T_j and, as our experiments indicate, provide a very good estimate of the similarity between two candidate test stimuli.

Once the hardest to detect fault vector $\underline{f}(T_j)$ for test stimulus T_j is computed, we can then determine the corresponding fault coverage, test access and sampling delay as described in the following section.

6. Evaluation of test fitness

Given the fault vector $\underline{f}(T_j)$ and the corresponding measurement values, evaluating the fitness of test stimulus T_j consists of two distinct steps. First, based on the above information, the optimal set of test nodes are selected such that maximum number of faults in $\underline{f}(T_j)$ are detected (or isolated) with minimum average sampling delay. The next step consists of the actual evaluation of the fitness function.

Selection of test nodes: The algorithm proposed for optimal test node selection is as following. Let us assume that F_p denotes the number of faults detected at test node C_p with average sampling delay S_p. The test nodes $\{C_p,\ p = 1 \text{ to } N_C\}$, where N_C is the total number of test nodes, are then ranked according to the increasing values of the product $(F_T\text{-}F_p).S_p$, where F_T is the total number of faults detected by test stimulus T_j. From this ranked list, test nodes with lowest values of $(F_T\text{-}F_p).S_p$ are chosen one at a time till the number of faults detected by the selected set of test nodes equals the total number of faults detected by test stimulus T_j, i.e., F_T. The number of total test nodes thus selected is denoted by N_T and the average sampling delay is denoted by S.

Evaluation of fitness function: The function used to compute the fitness of each test stimulus T_j must take into account the objective of the proposed test generator, which is to detect (or isolate) the maximum number of faults with minimum test access and minimum sampling delay. Therefore, a good heuristic for the fitness function will be of the following form:

$$a \cdot D(\underline{f}(T_j)) / \left|\underline{f}(T_j)\right| - b \cdot N_T / N_C - c \cdot S / L \qquad \text{(EQ 1)}$$

where, $D(\underline{f}(T_j))$ denotes the number of hardest to detect faults detected by T_j and L denotes the total duration of the test stimulus. The non-negative weights a, b and c depend on the relative importance of the three sub-objectives, i.e., maximizing fault coverage, minimizing test node and minimizing sampling delay. For example, if the goal was to maximize fault coverage without any constraint on test access or sampling delay, then the corresponding weights will be $a = 1$, $b = 0$ and $c = 0$. For our experimentation, we have assumed $a = 0.7$, $b = 0.2$ and $c = 0.1$. Note that, the value of the fitness function drives the test generation and optimization process described in Section 4.

7. Experimental results

To illustrate our methodology, let us consider the leapfrog filter shown in Figure 1. We assume that we are allowed to apply a transient test stimulus with a maximum frequency of 10 kHz up to a time duration of 2 ms. In practice, the frequency and the duration of the test stimulus will be determined by the tester specifications. The test stimulus was constructed as a piece-wise linear waveform with 20 segments and the maximum peak-to-peak voltage allowed was from -15 V to 15 V at steps of 1 V. Note that, the power supply voltage levels for the leapfrog filter were -15 V and 15 V respectively. We also assume that at most 50% of the circuit nodes can be probed at for testing purposes. This is a realistic assumption because of limited test access in most practical circuits. The netlist of the leapfrog filter circuit was coded in Spectre format and SpectreHDL behavioral models were used for the op-amps in Figure 1. The faults in the fault list were specified as percent deviations of the component parameters and Spectre was used as the core simulator during fault simulation.

Case study 1: For our first experiment, we generate tests to detect a set of 72 parametric faults in the leapfrog filter shown in Figure 1. The detection threshold (Section 2.1.) is assumed to be 0.2 V, and the tests are generated twice, once using full-circuit fault simulation and once using the partial fault simulation technique described in Section 5. Figure 6 shows the test stimuli and the resulting node voltage measurements for some of the faults in the given fault list. Full-circuit fault simulation was used to generate the test stimulus in this case.

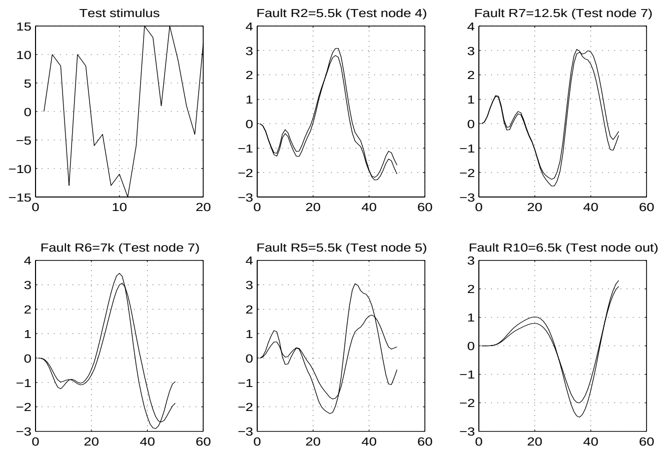


Figure 5. Test stimulus with full simulation

Similarly, Figure 6 shows the test stimuli generated using partial simulation and the corresponding node voltage measurements for the same subset of faults.

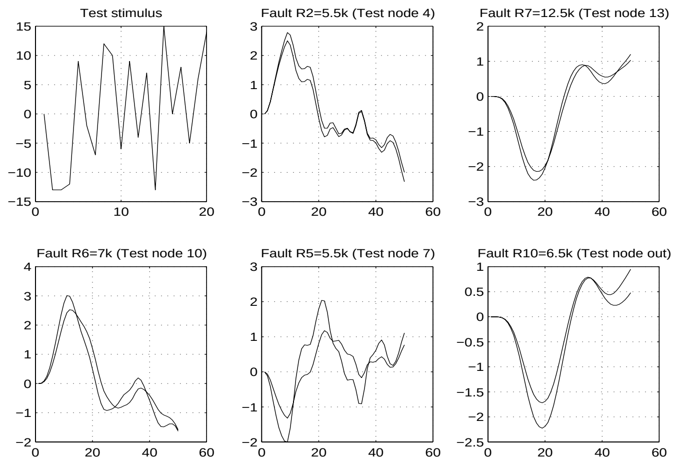


Figure 6. Test stimulus with partial simulation

Table 1 shows the comparison between test generation using full-circuit simulation and the same using partial simulation. Note that, the CPU time (on Sun Ultra-1) required by the latter is only 35% of that required by the former. Also note that, the fault coverage is the same (100%) in both cases, although the test access and the average sampling delay are slightly different.

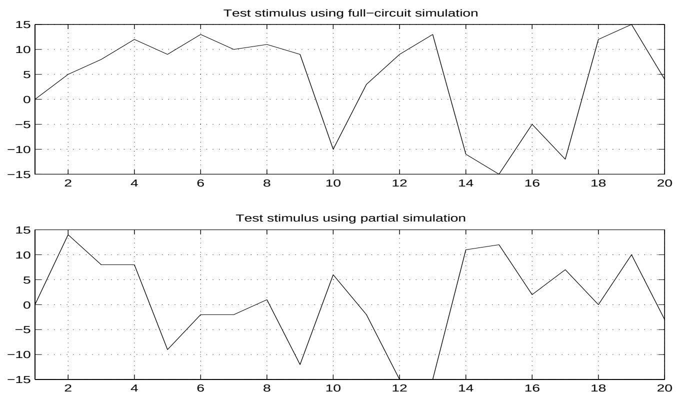


Figure 7. Test stimuli for state variable filter

TABLE 1. Test generation data for leapfrog filter

	Fault coverage	Test access	Sampling delay	CPU time
Full-circuit simulation	100%	28%	0.03 ms (average)	2138s
Partial simulation	100%	35%	0.02 ms (average)	756s

Case study 2: In our second experiment, we generate tests for detecting 60 parametric faults in the state variable filter shown in Figure 8 with a detection threshold of 0.5 V.

Figure 7 shows the test stimuli generated by using the two different fault simulation strategies. The corresponding comparison data are shown in Table 2.

TABLE 2. Test generation for state variable filter

	Fault coverage	Test access	Sampling delay	CPU time
Full-circuit simulation	100%	18%	0.08 ms (average)	1157s
Partial simulation	100%	45%	0.09 ms (average)	532s

Case study 3: For our final case study, we generate tests for isolating faults down to the corresponding sub-circuits within the

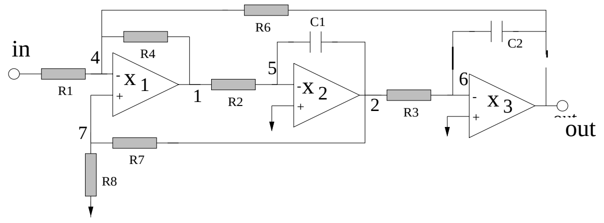
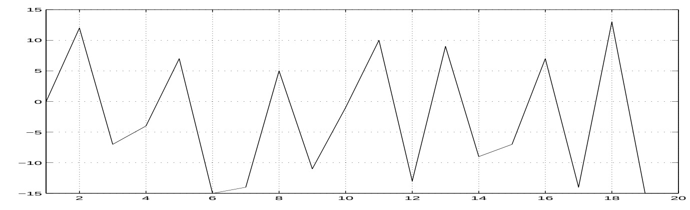
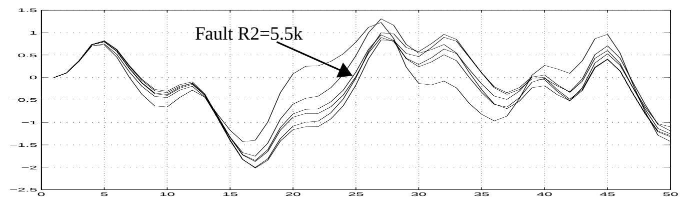


Figure 8. State variable filter

leapfrog filter as shown in Figure 1. Figure 9 shows the test stimulus and a few of the corresponding node voltage measurements (at test node 5) that isolates the fault R2 = 5.5 k from all other faults belonging to the remaining *five* partitions. The CPU time required to generate tests for isolating all the 72 faults (considered in case study 1) is 2149 s. The threshold used to differentiate between faults is 0.2 V.



(a) Test stimuli



(a) Measurement at test node 5

Figure 9. Fault isolation in leapfrog filter

8. Conclusions and future work

In this paper, we have presented a novel transient test generation methodology for analog circuits. The proposed approach, to the best of our knowledge, is the only technique that determines the optimal test stimuli as well as the optimal set of test nodes and sampling instants for both fault detection and isolation in analog circuits. It also takes into account the effects introduced by component tolerances and measurement inaccuracy. One key feature of the test generator is that it accommodates the hardware limitations of a real tester and can generate test waveforms that are suited for generation on a tester. The proposed approach can be applied to both linear and non-linear circuits as it uses a standard circuit simulator (Spectre) and behavioral models for fault simulation. The test development time is very reasonable for circuits with moderate complexity. This is more so keeping in mind that test development is a one-time process. Finally, the test generator can be easily modified so that the generated test waveform is of a particular type, e.g., piece-wise linear or step waveform, etc. This offers greater flexibility to the testing process.

Future work will involve extending and applying the proposed test generator to mixed-signal and large analog circuits.

9. References

[1] M. J. Marlett and J. A. Abraham, “DC-IATP: An Iterative Analog Circuit Test Generation Program for Generating DC Single Pattern Tests,” *Proc. Int’l Test Conference*, 1988, pp. 839-845.

[2] L. Milor and V. Visvanathan, “Detection of Catastrophic Faults in Analog Integrated Circuits,” *IEEE Trans. on CAD*, Vol. 8, 1989, pp. 114-130.

[3] N. B. Hamida and B. Kaminska, “Analog Circuit Testing based on Sensitivity Computation and New Circuit Modeling,” *Proc. Int’l Test Conference*, 1993, pp. 331-343.

[4] M. Slamani, B. Kaminska and G. Quesnel, “An Integrated Approach for Analog Circuit Testing with Minimum Number of Detected Parameters,” *Proc. Int’l Test Conferenc*e, 1994, pp. 631-640.

[5] S. Tsai, “Test Vector Generation for Linear Analog Devices,” *Proc. Int’l Test Conference*, 1991, pp. 592-597.

[6] G. Devarayanadurg and M. Soma, “Dynamic Test Signal Design for Analog ICs,” *Proc. Int’l Conference for CAD*, 1995, pp. 627-629.

[7] H. H. Zheng, A. Balivada and J. A. Abraham, “A Novel Test Generation Approach for Parametric Faults in Linear Analog Circuits,” *Proc. VLSI Test Symposium*, 1996, pp. 470-475.

[8] S. Chakrabarti and A. Chatterjee, “Diagnostic Test Pattern Generation for Analog Circuits Using Hierarchical Models,” *Proceedings, International Conference on VLSI Design*, January 1999, pp. 518-523.

[9] R. Voorakaranam, S. Chakrabarti, J. Hou, A. Gomes, S. Cherubal, A. Chatterjee, W. Kao, “Hierarchical Specification-Driven Analog Fault Modeling for Efficient Fault Simulation and Diagnosis,” *Proc. Int’l Test Conference*, 1997, pp. 903-912.

[10]H. Walker, S. W. Director, “VLASIC: A Catastrophic Fault Yield Simulator for Integrated Circuits,” *IEEE Trans on CAD*, Vol. CAD-5, No. 4, October 1986, pp. 541-546.

[11]A.V. Gomes, R. Voorakaranam and A. Chatterjee, “Modular fault simulation of mixed signal circuits with fault ranking by severity,” *Proc. IEEE Int’l Symposium on Defect and Fault Tolerance in VLSI Systems*, 1998, pp. 341-348.

[12]J.W. Bandler and A.E. Salama, “Fault Diagnosis of Analog Circuits”, *Proc. IEEE*, Vol. 73, August 1985, pp. 1279-1325.

[13]J. Hou and A. Chatterjee, “CONCERT: A Concurrent Fault Simulator for Analog Circuits,” *Proc. Int’l Conference on CAD*, 1998, pp. 384-391.

[14]P. M. Lin and Y. S. Elcherif, “Analog Circuits Fault Dictionary - New Approaches and Implementation,” *Circuit Theory and Applications*, 1985.