New Techniques for Failure Analysis and Test Program Design

by
C.E. Hymowitz, L.G. Meares, B. Halal
Intusoft P.O. Box 710, San Pedro CA 90733-0710, info@intusoft.com

Abstract - This paper discusses a currently proposed technique (sensitivity analysis) for analog fault analysis and describes several new software techniques already in use to perform analysis, diagnosis, and isolation of failures in analog and mixed-signal circuits and systems. Unique methods and algorithms for schematic entry, setting of failure characteristics, definition of test strategies, recording of simulation-based measurements, reduction of time-constrained simulation problems, creation of fault trees, and test sequencing are all discussed.

I. Introduction

The reasons for improving the analog and mixed-signal test program set design and Failure Mode Effects Analysis (FMEA) process are well documented [1-7]. For instance, identification of chronic production faults, improved safety and reliability through the analysis of difficult to test failure modes, investigation of circuit failure mechanisms, and excessive part stress could all benefit from improved simulation software. Yet few specialized software products currently exist to help define a structured approach to analog and mixed-signal circuit test procedures.

Testing is generally done to assure product performance, but a specialized software tool can add benefits in other ways. First, if it can identify failure modes that aren't tested, the designer will have found either unnecessary parts or a flaw in the acceptance test. In both cases, the quality of the product would be improved. Another important task is the tracking of component quality during production. Frequently, a product will evolve as new revisions are introduced and the product's performance will drift away from its design center. This deviation is observable from the acceptance test results, but software could also allow the designer or test engineer to track the nearby failures, including parametric failures. Lastly, a common design and test development tool would be useful in bridging the link, and frequently the information gap, between the design and test engineering departments.

II. Problems With Sensitivity Based Failure Analysis Techniques

When SPICE performs a simulation, it uses a matrix formulation of the form: $[G] * [V] = [I]$ where $[G]$ is an admittance matrix, $[V]$ are the node voltages and $[I]$ is the current matrix. The details of the implementation augment these matrices to account for the voltage sources. It turns out that a very simple mathematical transform can yield the sensitivity of any element in $[V]$ with respect to every parameter in $[G]$ by performing only one additional matrix computation involving the adjoint of the matrix $[G]$, $[G]^a$. This procedure is described in greater detail in reference [8], chapter 6. This corresponds nicely to the usual circuit analysis problems where just a few signals, usually outputs, need to be characterized. Compare this to an approach that solves the sensitivity problem by perturbing each parameter. Clearly, the adjoint method is superior.

Sensitivity can be used to compute the combined effect of all tolerances on a given output. The result can be used in several ways. First the RSS method computes the tolerance as the square root of the sum of the product of the sensitivity and the tolerance squared; that is, $tol(v) = Sqrt(\Sigma j\ (tol(j) * dv/dh_j)^2)$. For linear circuits, the result turns out to be the same as the tolerance for a Monte Carlo analysis.

The second use of sensitivity is to perform extreme value analysis, EVA. In certain cases, this is the same as a worst case, but as we will see, there is no guarantee. For extreme value computation, the sign of the outputs' derivative will be used to tell us which direction to move each parameter to make the parameters move the output to its maximum or minimum value. Then 2 analyses can be performed by first inserting the tolerance extremes that produce a maximum variation, and then inserting the ones that produce the minimum value. By extreme value, we refer to the parameter extremes, not the output. If a second sensitivity analysis is done at the extremes, the sign of the sensitivities are frequently opposite their nominal value, indicating that a worst case solution is found using some parameter values that are not at their extremes. This result occurs with sufficient regularity to cast doubt upon the benefits of extreme value analysis. However, even when the sign of the derivatives are the same, there is no guarantee that the worst case value has been found.

When tolerances become large, the sensitivity analysis obviously fails for digital and mixed signal circuits but it can also fail for linear circuits. To account for large changes in sensitivity, more work has to be done. Fortunately, the work only grows in proportion to the number of parameters that change by a large amount, using the techniques in reference [8], chapter 8. Again, however, nonlinear circuits invalidate the results. Several examples can be found in references [2,17].

Both RSS and EVA analyses are used to save computational resources. In certain cases, they produce the same results as Monte Carlo analysis. For nonlinear circuits, these cases can only be found by running the Monte Carlo analysis; this defeats the purpose of using RSS or EVA at all.

Failure analysis is an even more extreme case, where catastrophic failures require a MORE complex large parameter change approach which still breaks down unpredictably for nonlinear circuits. Again, the approach saves resources but can't be validated for nonlinear circuits without consuming the resources that it saved.

For fault isolation, it's useful to consider all test points and circuit quantities in the system in order to be able to select the best ones. When this "overkill" in analysis is performed, the computational savings vanish, and in fact, the number of Monte Carlo simulations is frequently less than the number of test points for which tolerance are required, thereby making it the best choice for conserving computational resources.

In references [9,10] claims are made that failures can be inferred from tolerances using sensitivity analysis. To accomplish this, a product specification is required. Based on that specification, tolerances are extracted using sensitivity analysis. The authors claim that they can detect out-of-tolerance failures based upon these data. The underlying assumption is that test measurements are linearly related to parameter changes. **This is simply not true for most circuits.** The best use for sensitivity analysis is to give the circuit designer insight; pointing him/her towards potential problem areas. Extrapolating sensitivity through nonlinearities is ill-advised.

III. A Robust Approach To Analog And Mixed-Signal Test

Test design demands a large database of faulty circuit behavior. The only practical method of gathering the data in a reasonable time frame is via simulation. With this in mind, a failure analysis and test development tool has been created based upon existing technology from two areas; the analog and mixed-signal world of SPICE simulation [11,12] and inference modeling [5,6]. The new tool is tailored to the specialized needs of the test and reliability engineer, but is also useful for the design engineer who must initiate the test development process. The tool provides an interactive design environment for the synthesis of diagnostic tests, generation of fault dictionaries, and the building of diagnostic fault trees.

Since the category of analog and mixed-signal test synthesis and sequencing software is relatively new, a number of unique techniques were developed to solve key parts of the FMEA (failure mode effects analysis) and test program set design process.

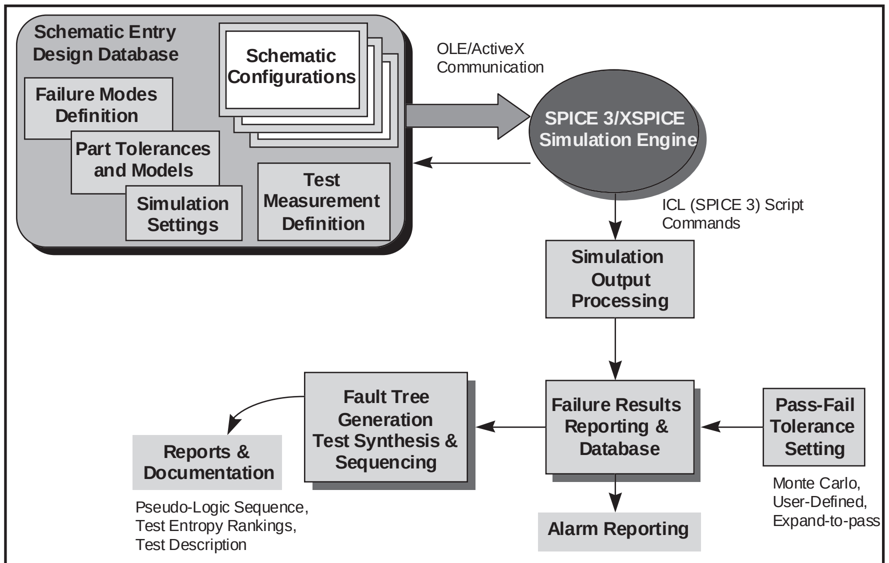

Schematic Entry Design Database
Schematic Configurations
Failure Modes Definition
Part Tolerances and Models
Simulation Settings
Test Measurement Definition
OLE/ActiveX Communication
SPICE 3/XSPICE Simulation Engine
ICL (SPICE 3) Script Commands
Simulation Output Processing
Fault Tree Generation Test Synthesis & Sequencing
Failure Results Reporting & Database
Pass-Fail Tolerance Setting
Reports & Documentation
Pseudo-Logic Sequence, Test Entropy Rankings, Test Description
Alarm Reporting
Monte Carlo, User-Defined, Expand-to-pass

Figure 1, A diagram of the software system implemented to provide automated failure analysis and test program set development.

The software, outlined in figure 1, provides the aforementioned benefits. The schematic entry program is specially enhanced to hold the entire design database, including part and model values and tolerances, and all part failure mode characteristics. It also contains a description of the various test configurations and measurements for each test.

The failure analysis process begins by simulating the circuit's or system's behavior with a single failure mode inserted. The schematic builds the required netlist for IsSpice4, a SPICE 3/XSPICE based analog and mixed signal simulator [13-15], using Windows-based ActiveX communication. The simulation output data is processed using the Berkeley SPICE 3 Interactive Command Language (ICL) in order to automatically extract the desired measurements from the simulation results. This process is continued automatically, one fault at a time, until all of the faults are simulated. The measurements are then parsed into various report forms which contain user-defined test limits. This database becomes the Fault Dictionary. Finally, using the fault dictionary data, tests are sequenced into a fault tree.

IIIa. Configurable Schematics

A long-standing problem in electrical and mechanical circuit design has been the conflict between the needs of the designer and the needs of production and manufacturing. The main vehicle for conveying the circuit specification is

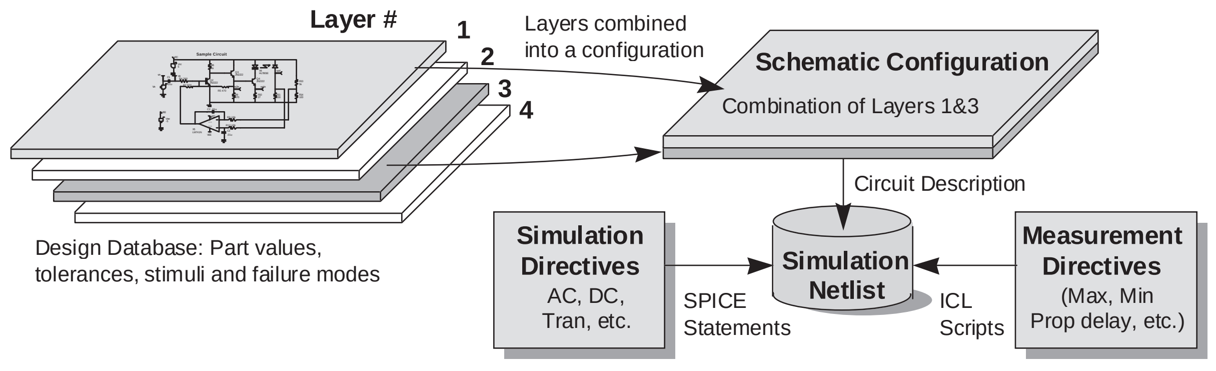


Figure 2, A unique reconfigurable schematic tool allows different schematic layers to be combined in order to create various circuit configurations. Simulation and measurement directives are then added to create multiple test descriptions which are simulated for each and every fault in the system.

the schematic diagram which is used to describe the circuit topology as well as the details of how production will build the hardware.

The designer is concerned with creating a circuit that meets specifications. This is done chiefly through various EDA tools, but mainly with circuit simulation. The designer must build multiple test configurations, add parasitic components and stimuli, and even include system elements in the simulation. A top-down design methodology, where different levels of abstraction are inserted for different components, is commonplace. Modeling electrical behavior often results in different representations for different test configurations. In general, the schematic becomes so cluttered with circuitry and data, that it must be redrawn for production, greatly increasing the probability of a transcription error.

It should be noted that simulation can and should be used to check out the test setup. Using simulation to evaluate and design test jigs speeds up the overall process and doesn't consume precious time on the Automatic Test Equipment (ATE).

The need for a reconfigurable schematic capability becomes even more mandatory when we analyze the needs of the test program development engineer. In order to be effective, the simulation process can not become burdened with the bookkeeping intricacies of multiple test fixtures and settings. The designer must have a way to connect various stimuli and loads to core circuitry and to group the desired SPICE analyses and test measurements with each schematic configuration.

Until now, the best approach has been to hide these special configurations within subcircuits. While this approach works for hierarchical schematic entry, it doesn't solve the problem of adding test equipment, different stimulus inputs, or dealing with multiple simulation scenarios in a reasonable fashion.

A test setup provides loads, voltage and current stimuli and instrumentation connections at specific points on the Unit Under Test (UUT). When viewed in a broader context, the combination of the test setup circuitry and the UUT can be considered to be a circuit configuration in and of itself. Indeed, for simulation purposes, the test setup circuitry must be included as part of the circuit. Most Test Program Sets (TPSs) implement multiple setups during the testing sequence. This increases the simulation burden by requiring a separate schematic for every test setup.

The system described by figure 2 addresses the multiple test setup problem with a unique solution. It allows the user to assign each setup/UUT combination a different configuration name and simulates all of the stand-alone configurations in a batch operation. The setup/UUT combination,

Table 1, SPICE3 Syntax for Various Failure Modes

Fault	Before Insertion	After Fault Insertion
Shorted Base Emitter	Q1 12 19 24 QN2222A	Q1 12 19 24 QN2222A
		Rshort_19 19 24 .1
Open Resistor	R3 17 0 10K	R3 17_open 0 10K
		Ropen_17 17_open 17 100Meg
Low Beta	Q1 12 19 24 QN2222	Q1 12 19 24 Q1_Fail
Parametric fault	.MODEL QN2222 NPN AF=1 **BF=105**	.MODEL Q1_Fail NPN AF=1 **BF=10**
	BR=4 CJC=15.2P CJE=29.5P…	BR=4 CJC=15.2P CJE=29.5P…
Resistor Stuck	R1 6 0 1K	R1 6 0 1K
2V below Vcc		Rstuck_6 6_Stuck 6 10.00000
		Bstuck_6 6_Stuck 0 V= Vcc- 2
Time Dependent	L2 3 0 62U	L2 3 0 62U
Inductor Fault		Rstuck_3 3_Stuck 3 10.00000
		Bstuck_3 3_Stuck 0 V=Time>10n ? 0 : V(3)

defined during the schematic entry process, is called a “circuit configuration”. Every circuit configuration is composed of one or more schematic layers. An active layer can be thought of as a transparency that overlays other transparencies such that as you view them, you see the complete circuit configuration schematic. Circuit nodes on the top layer connect with nodes on underlying layers as if the drawing were created on a single page. The schematic allows mixing and matching of layers to form the required circuit configurations. Any circuitry, elements, or documentation can be placed on any layer. While PCB layout software has had a similar feature for quite some time, a configurable schematic has not been implemented (to the best of our knowledge). This is the first known graphical entry method which is capable of solving the Test - Simulation bridge using a reconfigurable layered schematic approach.

IIIB. Failure Definition

Each component is defined by a set of nominal device and model parameter attributes, as well as parametric tolerances. Each component also has a set of defined failure modes. Component failures are well characterized by a finite number of catastrophic failure modes. Unusual failure modes get a lot of attention. For example, it is unusual to find a PNP transistor die in an NPN JANTXV package. Although these things do happen, they are very rare. If acceptance testing detects only 99% of all failed parts, then the quality of the product increases 100 fold after these tests are performed. For many products, the increased quality guarantees that products with undetected failures will not be delivered to the customer.

Process faults could cause the shift of many parameters simultaneously. When this is a consideration, as is usually the case for IC’s, the process parameters are monitored separately. If the process fails, the unit is rejected before the acceptance test is performed. Therefore, acceptance testing does not need to account for multiple parametric failure modes.

Initially, parts include the catastrophic failure modes as defined in the Navy’s CASS (Consolidated Automated Support System) Red Team Package[16]. However, a simple interface is included to allow users to edit the predefined failure modes or add their own catastrophic or parametric failure modes (trace shorts, IC bridging or interconnect failures, solder splashes, low beta, device coupling, etc.). The characteristics (open/short/stuck resistance) of each failure mode can be defined by the user.

Failure modes are simulated by programmatically generating the proper SPICE 3 syntax to describe the failure. Any node on a part can be shorted to any other node, opened, or stuck. The stuck condition allows the user to attach a B element expression. The B element is the Berkeley SPICE 3 arbitrary dependent source which is capable of analog behavioral modeling [11,12]. The expressions can refer to other quantities in the design such as nodes and currents, thus creating an unlimited fashion in which to “stuck” a node. A series of examples are shown in Table 1.

All of the failure mode definitions are carried out in a graphical manner. No script writing or programming is necessary in order to define or simulate a fault. There is no need for the user to write or know the required SPICE syntax, and the schematic is not altered when a fault is inserted.

IIIC. Measurement Definition

In order to create a “test”, the user must combine a circuit configuration with a set of SPICE simulation directives and a desired measurement which will be made on the resulting data. Therefore, the simulator, or a data post-processing program, must be available to extract and store information from each desired test point waveform. The former was chosen for this tool and implemented using ICL which is available in SPICE 3.

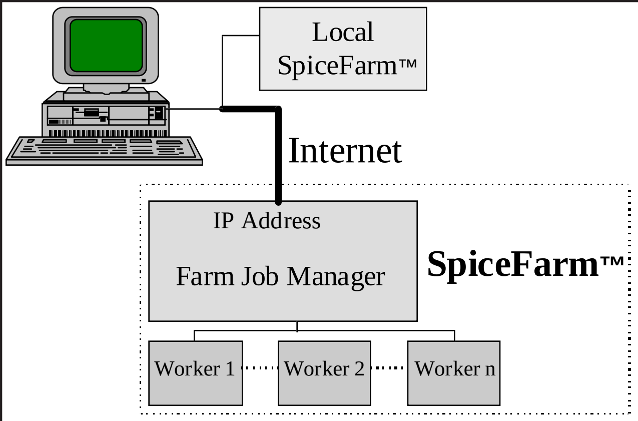


Figure 3, A farm of computers running SPICE simulations in parallel is used to dramatically reduce the required computational demands of failure analysis. The SpiceFarm can be made up of a bank of locally available computers or accessed over the Internet.

The software uses the IsSpice4 simulation engine to perform the failure analysis. IsSpice4 includes and expands upon the standard Berkeley SPICE 3 ICL. ICL is a set of commands that can direct SPICE to perform various operations such as running a particular analysis or changing a component value. The commands, which look and act like Visual Basic scripts, can be run interactively or in a batch mode. IsSpice4 contains ICL functions which allow SPICE 3 plots, or sets of vectors (i.e. waveforms) to be scanned and measured with imaginary cursors. In contrast to traditional SPICE "dot" statements, ICL commands are performed in order, one at a time. This makes ICL scripts perfect for describing test procedures.

A variety of functions are available for setting the cursor positions and for measuring and processing simulation results. As shown in figure 2, these measurement scripts are combined with traditional SPICE analysis directives and a test configuration description to form a simulatable IsSpice4 netlist. Virtually any measurement can be recorded on any voltage, current, or power dissipation waveform such as a maximum value, peak-peak value, rise/fall time, propagation delay, Iddq values, and Fourier coefficients.

IIID. Running The Failure Analysis

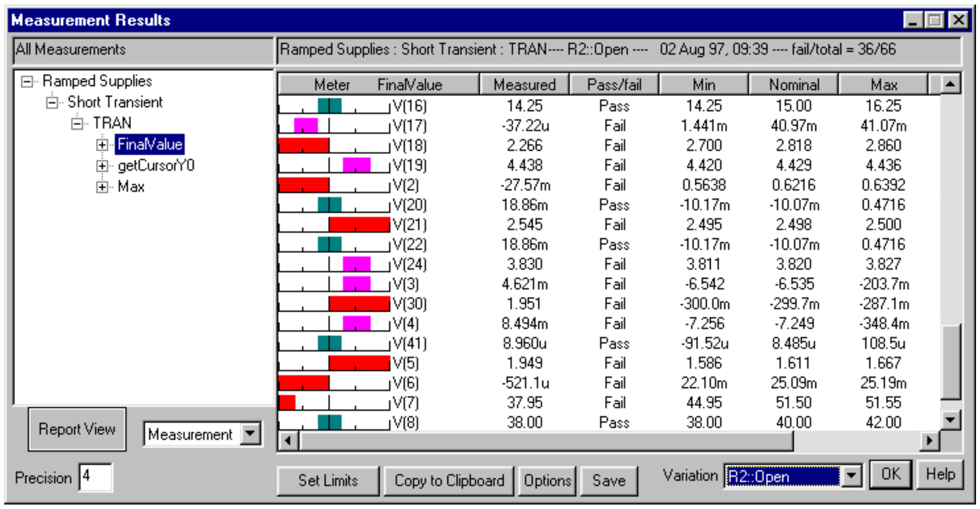
Before the failure analysis can begin, the test engineer will have to decide which tests to perform, how to set up each test, and the test limit boundaries. This last step can be time-consuming, especially if design specifications are not available. Therefore, special methods are available for setting limits on groups of tests using default tolerances, Monte Carlo simulation results or a unique "Expand to Pass" feature which pushes out the test limit boundaries. This last method allows the designer to easily account for variations in the environment or test equipment which might otherwise cause false failures on tests with tight tolerances.

Once the test limits are set, failure mode simulation can proceed one fault mode at a time or in an automatic fashion until all of the fault modes are simulated.

IV. Overcoming Simulation Runtime Issues

There are several ways to overcome simulation runtime issues associated with the large number of fault simulations that must be performed. The first is to use a faster computer or many fast computers to perform the runs more quickly and in parallel. In addition, test development for diagnostic cases usually proceeds along the path of functional testing. This is simply not mandatory, or the most efficient path, in most cases. The test engineer is limited by his knowledge of the circuit operation which is most easily revealed through functional testing. But for diagnostic test development, simpler test methods can be employed which greatly reduce simulation runtimes [2].

Past work [9,10] implies that simulation runtime is a major inhibitor to the proposed methodology. However, these remarks tend to ignore recent developments in the area of model optimization and behavioral modeling, Analog Hardware Description Language (AHDL) modeling, and current simulator and computer performance.



Ramped Supplies : Short Transient : TRAN---- R2::Open ---- 02 Aug 97, 09:39 ---- fail/total = 36/66

Meter	FinalValue	Measured	Pass/fail	Min	Nominal	Max
🟩	V(16)	14.25	Pass	14.25	15.00	16.25
🟪	V(17)	-37.22u	Fail	1.441m	40.97m	41.07m
🟥	V(18)	2.266	Fail	2.700	2.818	2.860
🟪	V(19)	4.438	Fail	4.420	4.429	4.436
🟥	V(2)	-27.57m	Fail	0.5638	0.6216	0.6392
🟩	V(20)	18.86m	Pass	-10.17m	-10.07m	0.4716
🟥	V(21)	2.545	Fail	2.495	2.498	2.500
🟩	V(22)	18.86m	Pass	-10.17m	-10.07m	0.4716
🟪	V(24)	3.830	Fail	3.811	3.820	3.827
🟪	V(3)	4.621m	Fail	-6.542	-6.535	-203.7m
🟥	V(30)	1.951	Fail	-300.0m	-299.7m	-287.1m
🟪	V(4)	8.494m	Fail	-7.256	-7.249	-348.4m
🟩	V(41)	8.960u	Pass	-91.52u	8.485u	108.5u
🟥	V(5)	1.949	Fail	1.586	1.611	1.667
🟥	V(6)	-521.1u	Fail	22.10m	25.09m	25.19m
🟥	V(7)	37.95	Fail	44.95	51.50	51.55
🟩	V(8)	38.00	Pass	38.00	40.00	42.00

Figure 4, A sample dialog showing the pass-fail results of a failure simulation with an open resistor. The left side of the tree shows a list of the measurements to be performed. On the right side are the test results and limits. At the bottom (R2:open) any fault in the system can be selected using the drop-down list and its effects reviewed.

A newly developed method uses a "farm" of computers. Referring to figure 3, the user has control over whether the simulations are run on the local computer, a local set of computers, or at a remote site housing a bank of fast computers. A farm job manager appears as an IP address to the user and receives the simulation jobs over the Internet. The simulations are parceled out to local "workers". The results are sent back to the user for display as the jobs are finished. The data sent between the user's computer and the farm is encrypted, greatly easing security concerns.

The concept of a farm of computers has been implemented by the software developer, Intusoft, and is called the SpiceFarm™.

V. Results Reporting

The results for each failure simulation are reported in a special dialog, as shown in figure 4. A tree list lists each drawing configuration, the simulation setups, the individual analyses in each simulation, and the user-defined measurements. The measurements are made and any number of test points (i.e. waveforms). The results of each test are displayed on the right. There are two other display types which are not shown; one that shows the results of a single test for all of the failure modes, and another that shows a histogram of a test measurement vs. all failure modes.

The meter on the left center of the report is used as a quick indicator of the measurement's pass-fail status. A long bar extending to the left or right of the meter center indicates that the associated failure mode moves the measured value outside of the pass/fail limits by more than 3 times the difference between the upper and lower pass/fail limits (e.g., a very high probability of failure detection). A short bar indicates that the failure is detected but is out of limits by less than one tolerance range (e.g. could be an uncertain detection and may merit further investigation using Monte Carlo techniques). Together, these results dialogs constitute the fault dictionary and may be used to populate various third party reliability tools with failure analysis data that is normally otherwise generated with guesses, intuition, or breadboard derived values.

VI. Synthesizing Tests

The process of fault tree generation takes place in the Fault Tree design dialog (figure 5) using a novel test sequencing technique [1].

It is generally accepted that the best fault tree is one that arrives at the highest probability failure conclusion with the least amount of work. The best test is then the test that produces the optimum fault tree, free from errors. Several methods for selecting the best test out of those remaining have been proposed [3,4]. Given an equal probability of occurrence for each fault, the best test is usually the one that evenly divides the input group between the pass and fail group. A somewhat more complex procedure has also been proposed. Note that a general solution, made by exhaustive search, rapidly becomes intractable [4].

Tests are used to detect faults using the logic illustrated in figure 6. Each test has one input and 2 outputs. The input contains a list of failure modes, and the test performs the logic which is necessary to classify the outcome as pass or fail. Each outcome has a list of failure modes (ambiguity groups) that can be passed-on to successive tests. The process of selecting the best test in each ordered group results in a binary fault tree. After selecting a test, successive tests are placed on the pass and fail nodes of the tree until no more useful information is gained.

If the component failure rate is used to weight each fault in the ambiguity group, then we can assign a probability of detection to the pass group, p, and the fail group, q. What is really determined is the probability that a test will pass (p) and the probability that a test will fail (q). Then p + q = 1.0, because the test will either pass or fail. The input probability must be 1.0 because one of the conclusions in the current ambiguity group will be the answer. Weighting is used to reassess individual failure probability, given that the group is the answer at this point in the isolation process. Now the probability of reaching each conclusion can be predicted, based upon failure weights. The best fault tree can now be defined as the one which arrives at each failure conclusion with the least amount of work. If each test is equally difficult, then the work is the summation of the probabilities of reaching each conclusion. To compute these probabilities, we simply traverse the tree for each conclusion, multiplying the probabilities of each successive outcome, and then summing the resultant probabilities for each

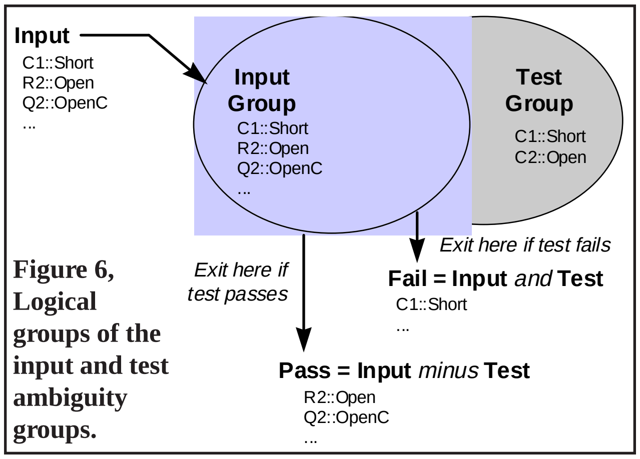
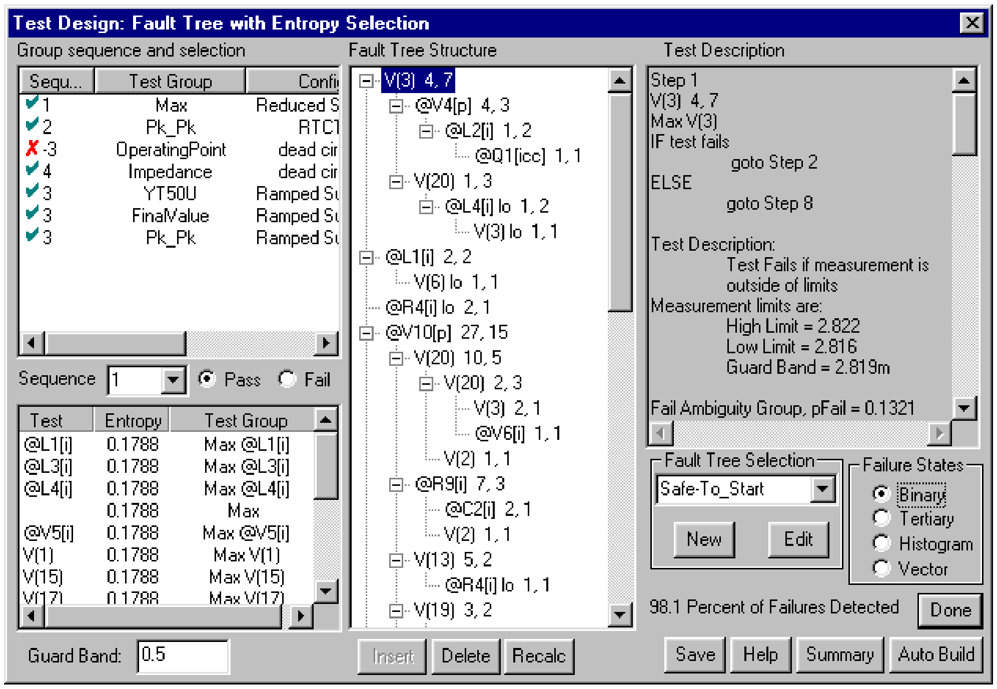


Figure 6, Logical groups of the input and test ambiguity groups.



Group sequence and selection

Sequ...	Test Group	Confi
✓1	Max	Reduced S
✓2	Pk_Pk	RTC1
✗-3	OperatingPoint	dead cir
✓4	Impedance	dead cir
✓3	YT50U	Ramped Su
✓3	FinalValue	Ramped Su
✓3	Pk_Pk	Ramped Su

Sequence 1 [✓ Pass] Fail

Test	Entropy	Test Group
@L1[i]	0.1788	Max @L1[i]
@L3[i]	0.1788	Max @L3[i]
@L4[i]	0.1788	Max @L4[i]
	0.1788	Max
@V5[i]	0.1788	Max @V5[i]
V(1)	0.1788	Max V(1)
V(15)	0.1788	Max V(15)
V(17)	0.1788	Max V(17)

Guard Band: 0.5

Fault Tree Structure

```
V(3) 4, 7
  @V4[p] 4, 3
    @L2[i] 1, 2
      @Q1[icc] 1, 1
  V(20) 1, 3
    @L4[i] lo 1, 2
      V(3) lo 1, 1
@L1[i] 2, 2
  V(6) lo 1, 1
@R4[i] lo 2, 1
@V10[p] 27, 15
  V(20) 10, 5
    V(20) 2, 3
      V(3) 2, 1
      @V6[i] 1, 1
    V(2) 1, 1
  @R9[i] 7, 3
    @C2[i] 2, 1
    V(2) 1, 1
  V(13) 5, 2
    @R4[i] lo 1, 1
  V(19) 3, 2
```

Test Description

```
Step 1
V(3) 4, 7
Max V(3)
IF test fails
        goto Step 2
ELSE
        goto Step 8

Test Description:
        Test Fails if measurement is
        outside of limits
Measurement limits are:
        High Limit = 2.822
        Low Limit = 2.816
        Guard Band = 2.819m

Fail Ambiguity Group, pFail = 0.1321
```

Fault Tree Selection: Safe-To_Start

Failure States: [✓ Binary] Tertiary Histogram Vector

98.1 Percent of Failures Detected

Figure 5, The following dialog is used to sequence tests into a fault (diagnostic) tree. The resulting test sequence can be used for fault isolation or a product acceptance test. The Fault Tree structure shows the test and the number of faults in the ambiguity groups (pass first, then fail). For instance, L4[I]lo 1, 2 would have 1 fault in the pass ambiguity group and 2 in the fail ambiguity group. The description of each highlighted test is shown to the right.

conclusion. The best result from this procedure is 1.0. The figure of merit tells us how well we did, and can be used to compare fault trees. Clearly, we must select tests which produce high probability outcomes. To find these tests, we compute the entropy of each useful test that is available: Entropy = $-p*\log(p) - q*\log(q)$

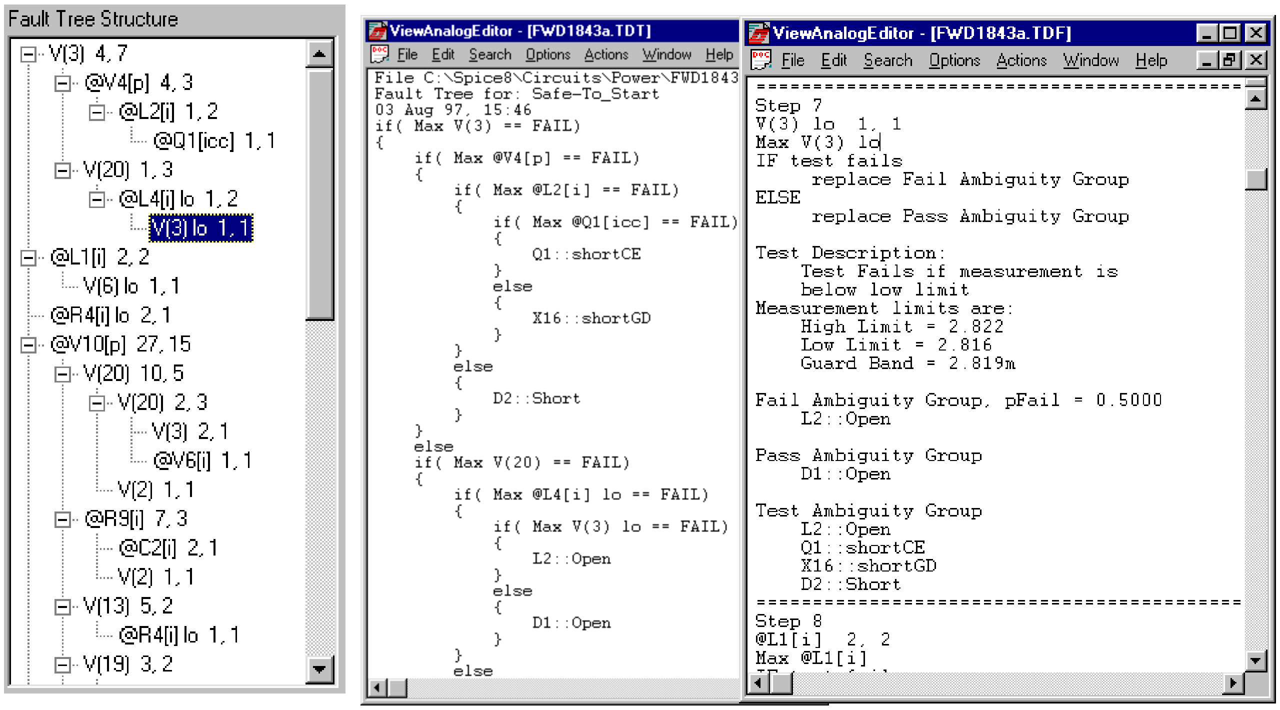
According to information theory, the highest entropy test contains the most information. Proceeding in this manner tends to produce efficient diagnostic or product acceptance trees. Tests may be grouped into a common pool, or group, for selection. This allows tests to be ordered not only by difficulty, but also by logical requirements; for example, high temperature vs. low temperature and safe-to-start vs. operating point.

Tests have only 2 outcomes, pass or fail; but a measurement can be compared with many different limits, creating a large number of possible tests for each measurement. For example, a binary measurement passes if the result is within the test limits, and fails if it is outside of the limits. A tertiary measurement is divided into 3 states; fail low, pass and fail high.

Ambiguity groups are created for each test. Test ambiguity groups contain a list of faults that can be detected; that is, faults that will be reported to the fail outcome it they are in the input fault group. The subset of the fault universe that goes into a test must be present in either the pass or fail outcome.

Measured values can migrate across pass-fail boundaries because of component and test tolerances. In order to produce robust tests, the concept of a guard band has been added to the test sequencing process. The guard band is measured in pass tolerance units and is used to eliminate from consideration any test that has fault detections within the guard band limit of the test boundary. Setting the test limit to the center of the guard band produces a test that is least likely to give false results.

Referring to figure 5, several groups of tests are available. They are shown in the Group Sequence


Fault Tree Structure
ViewAnalogEditor - [FWD1843a.TDT]
File Edit Search Options Actions Window Help
ViewAnalogEditor - [FWD1843a.TDF]
File Edit Search Options Actions Window Help

```
V(3) 4, 7
    @V4[p] 4, 3
        @L2[i] 1, 2
            @Q1[icc] 1, 1
    V(20) 1, 3
        @L4[i] lo 1, 2
            V(3) lo 1, 1
@L1[i] 2, 2
    V(6) lo 1, 1
@R4[i] lo 2, 1
@V10[p] 27, 15
    V(20) 10, 5
        V(20) 2, 3
            V(3) 2, 1
            @V6[i] 1, 1
        V(2) 1, 1
    @R9[i] 7, 3
        @C2[i] 2, 1
        V(2) 1, 1
    V(13) 5, 2
        @R4[i] lo 1, 1
    V(19) 3, 2
```

```
File C:\Spice8\Circuits\Power\FWD1843
Fault Tree for: Safe-To_Start
03 Aug 97, 15:46
if( Max V(3) == FAIL)
{
    if( Max @V4[p] == FAIL)
    {
        if( Max @L2[i] == FAIL)
        {
            if( Max @Q1[icc] == FAIL)
            {
                Q1::shortCE
            }
            else
            {
                X16::shortGD
            }
        }
        else
        {
            D2::Short
        }
    }
    else
    if( Max V(20) == FAIL)
    {
        if( Max @L4[i] lo == FAIL)
        {
            if( Max V(3) lo == FAIL)
            {
                L2::Open
            }
            else
            {
                D1::Open
            }
        }
        else
```

```
==========================================
Step 7
V(3) lo  1, 1
Max V(3) lo
IF test fails
     replace Fail Ambiguity Group
ELSE
     replace Pass Ambiguity Group

Test Description:
    Test Fails if measurement is
    below low limit
Measurement limits are:
    High Limit = 2.822
    Low Limit = 2.816
    Guard Band = 2.819m

Fail Ambiguity Group, pFail = 0.5000
    L2::Open

Pass Ambiguity Group
    D1::Open

Test Ambiguity Group
    L2::Open
    Q1::shortCE
    X16::shortGD
    D2::Short
==========================================
Step 8
@L1[i]  2, 2
Max @L1[i]
```

Figure 7, Several types of reports can be output. Shown above is a portion of the fault tree which points to the V(3) (voltage at node 3) test. The matching test description and hardware independent C-like pseudo-logic code to duplicate the fault tree are shown to the center and right, respectively.

and Selection section. Any of the test groupings can be activated using the Sequence drop-down list. The resulting ranking of the tests, in order of best to worst, is shown in the Test-Entropy section. All of the tests with the same sequence number will be evaluated at the same time. A fault tree can be sequenced from the activated tests manually, by hitting the insert button, or automatically, by hitting the AutoBuild button. This second action builds the entire fault tree and provides a percentage coverage value along with a summary of the remove/replace callout groups.

The test synthesis process culminates in the generation of pseudo-code to drive automatic test equipment (ATE). Two ATE independent formats are generated, describing the structure of the tests (figure 7); an Atlas-like version and a C code version.

VII. CONCLUSIONS

The simulation techniques employed here act as a "force multiplier" for development of diagnostics for analog and mixed signal circuits. In a relatively short time, compared with current methods, it was possible to analyze the failure characteristics of a design, generate a variety of tests, determine the failure mode detection characteristics of each test, and sequence these tests into an effective diagnostic fault tree.

Simulation has proven to be an effective method for identifying problem areas in fault detection. Simulation reveals key problem areas: first, by identifying the low probability fault detections for individual tests; and second, by providing an infrastructure for further circuit analysis when integration results do not agree with simulator predictions. Reasonable simulation times can be achieved by running multiple simulations in parallel using the "SpiceFarm" approach described herein. In addition, the new tools and techniques allow the engineer to study various circuit failure mechanisms, failure propagation paths and failure outcomes.

Accounting for the percentage of detection and isolation of failure modes is a difficult problem for the TPS developer. While accountability is easy to achieve in simple circuit analysis, it becomes considerably more difficult as mixed-signal systems are analyzed. The preceding techniques provide a variety of significant advantages.

REFERENCES

[1] Larry Meares, "Designing Tests Using Simulation", Evaluation Engineering Magazine, Parts I&II, April-May, 1998

[2] Charles Hymowitz, Larry Meares, Bill Halal, "New Techniques for Fault Diagnosis and Isolation of Switched Mode Power Supplies",1997 PCIM Conference

[3] Sharon Goodall, "Analog/Mixed Signal Fault Diagnosis Algorithm and Tool Review",1994 AutoTestCon Proceedings, pp. 351-359.

[4] W.R. Simpson and J.W. Sheppard, "Fault Isolation in an Integrated Diagnostic Environment, "IEEE D&T, Vol.10,No.1,Mar. 1993, pp52-66.

[5] C.Y. Pan and K.T. Cheng, "Test Generation for Linear, Time Invariant Analog Circuits, 3rd IEEE Intl. Mixed-Signal Testing Workshop, June 3-6, 1997.

[6] Harry H. Dill, "A Comparison of Conventional and Inference Model Based TPS Development Processes", AutoTestCon '95 Proceedings, pp 160-168.

[7] Harry H. Dill, "A Comparison of Conventional and Inference Model Based TPS Development Processes", 1997 AutoTestCon Proceedings.

[8] Jiri Vlach, Kishore Singhal, "Computer Methods for Circuit Analysis and Design", 2nd Ed., Van Nostrand Reinhold, 1994

[9] N.B. Hamida, K. Saab, D. Marche, B. Kaminska, and G. Quesnel, "LIMSoft: Automated Tool for Design and Test Integration of Analog Circuits", 2nd IEEE International Mixed Signal Testing Workshop, Quebec City, Canada, May 15-18, 1996.

[10] N.B. Hamida and B. Kaminska, "Analog Circuit Fault Diagnosis Based on Sensitivity Computation and Functional Testing", IEEE Design and Test of Computers, March 1992, pp.30-39.

[11] L.W. Nagel and D.O. Pederson, Simulation program with integrated circuit emphasis, ERL Memo No. ERL-M520, University of California, Berkeley, May 1975.

[12] B. Johnson, T. Quarles, A.R. Newton, D.O. Pederson, A. Sangiovanni-Vincentelli, SPICE 3F User's Guide, University of California, Berkeley, Oct. 1992.

[13] "IsSpice4 User's Guide", Intusoft, P.O. Box 710, San Pedro, CA 90733-0710, 1997.

[14] Fred Cox, W. Kuhn, J. Murray, S. Tynor, "Code-Level Modeling in XSPICE", Proceedings of the 1992 Intl. Syp. on Circuits and Systems, May 1992, IEEE 0-7803-0593-0/92.

[15] "XSPICE Users Manual", Georgia Institute of Research, Georgia Institute of Technology, Atlanta GA 30332-0800.

[16] CASS Red Team Package data item DI-ATTS-80285B, Fig. 1 - SRA/SRU Fault Accountability Matrix Table, pg 11.

[17] Intusoft Newsletter, Various Issues #51 Nov. 1997, #52 Feb. 1998, #53 April 1998, Copyright Intusoft 1986-1998