New Techniques for Fault Diagnosis and Isolation of Switched Mode Power Supplies

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 describes new software techniques to perform analysis, diagnosis, and isolation of failures in analog and mixed-signal circuits including switched mode power supplies. Unique methods and algorithms for schematic entry, setting of failure characteristics, definition of test strategies, recording of simulation based measurements, creation of fault trees and sequencing tests are all discussed. To ensure a realistic test of the new software techniques, diagnostics were developed for a moderately complex analog and mixed signal switched mode power supply. A discussion of some of the drawbacks of sensitivity based failure analysis techniques is also included.

I. INTRODUCTION

In late 1996, the SPICE simulation tool manufacturer Intusoft initiated development of a new product, tailored to the unique and demanding needs of the test engineer. This product, called Test Designer, provides an effective, 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.

The motivations for improving the analog and mixed-signal test set design and failure analysis process are plentiful and well documented [1-5]. For instance, identification of early production faults, improved safety and reliability through the analysis of difficult to test failures, investigation of power supply failure mechanisms such as power switch over-current, FET gate over-voltage, start-up failures, and excessive component stress could all benefit from improved simulation software. Yet little dedicated software currently exists to help streamline and organize analog and mixed-signal circuit test procedures.

Most testing is done to assure product performance standards. UL standards for various types of supplies require that tests be developed for overload protection circuitry, short circuit tests, and breakdown of components[6]. In some cases, when circuit analysis indicates that no other component or portion of the circuit is seriously overloaded as a result of the assumed open circuiting or short circuiting of another component, some tests can even be bypassed.

The software can add benefits in 2 other ways. First, if it can identify failure modes that aren't tested, you will have found either unnecessary parts or a flaw in the acceptance test. Either case would improve the quality of the product. Another important aspect is the tracking of component quality during the production lifetime. Frequently, a supplier's product will evolve or the supplier will be changed, and the product's performance will drift away from its design center. This drift itself is observable from the acceptance test results, but the software also allows you to track the nearby failures, including parametric failures. These, of course, aren't really failures in the sense that the product can't be shipped; rather, they are component quality indicators.

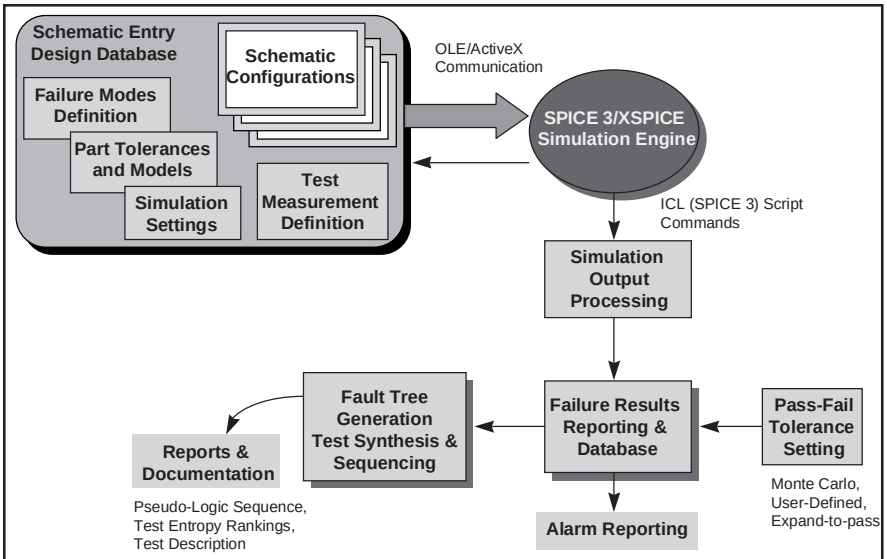


Figure 1, A block diagram of the software system implemented to provide automated failure analysis and test program set development.

The software, outlined in Figure 1, provides the aforementioned benefits and includes a complete system capable of design entry, simulation, analysis, and test synthesis and fault tree sequencing. The schematic entry program is specially enhanced to hold the entire design database including part and model values and tolerances, and all part failure modes. It also contains a description of the various test configurations and measurements for each test. Using Object linking and embedding (OLE) communication, the schematic builds the required netlist for IsSpice4, a SPICE 3/XSPICE based analog and mixed signal simulator. The simulation output data is processed using the Berkeley SPICE Interactive Command Language (ICL) in order to extract the desired measurements. This process is continued automatically until all of the faults are simulated. The measurements are then parsed into various report forms which contain the pass-fail tolerances. Finally, the tests are sequenced into a fault tree.

A six step process, described in detail below, is utilized for the development of fault diagnostics. In chronological order, the steps are:

- Design Entry including: Schematic Layers setup, Schematic Configurations setup, Simulation Directives setup, Failure modes characteristics definition, Test Configuration definition
- Measurement definition
- Pass-Fail tolerance setting
- Fault simulation
- Results reporting
- Failure states and sequencing

II. 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 method of conveying the designer's creation is the circuit schematic which is used to describe the behavior of the circuit 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 raising the probability of a transcription error.

The need for a reconfigurable schematic capability becomes even more mandatory when we analyze the needs of the failure analysis and test program development engineer. In order to be effective, the simulation process can not become burdened with the bookkeeping intricacies of multiple schematic variations and analysis specifications. 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 in subcircuits; for example a resistor's parasitic capacitance could be included in a subcircuit. While this approach works for hierarchical schematic entry and extending individual component models, it doesn't solve the problem of adding test equipment, different stimulus inputs, or dealing with multiple simulation scenarios.

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 is called a "circuit configuration". The circuit configuration is defined during the schematic entry process. 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 underly-

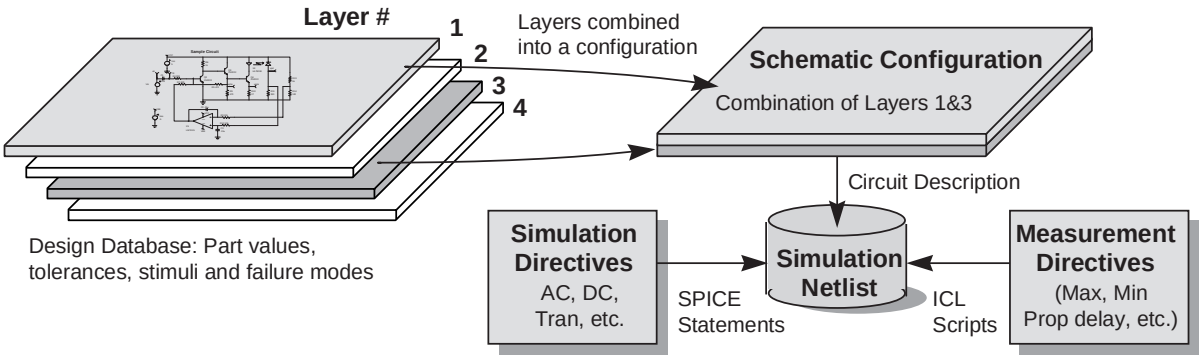


Figure 2, A unique reconfigurable schematic program allows different schematic layers to be combined in order to create various circuit configurations. Simulation and measurement directives are then added to create multiple tests descriptions.

ing 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.

Use of a layered concept in itself is not unique. It is generally used as a drawing management feature to remove complexity from the user's visual field, rather than making a multiplicity of configurations. 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.

III. SIMULATION DIRECTIVES

The system also allows different sets of SPICE analysis statements to be grouped and stored (Figure 2). For instance, an operating point analysis can be specified to run along with a brief transient analysis. In another group, a frequency response can be run with a long transient analysis. The user can then pair any set of SPICE simulation directives with any circuit configuration to create a unique "Test Configuration". For example, a single circuit configuration can be assigned multiple types of simulation analyses in order to define multiple test configurations.

IV. 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 associated failure modes. Initially, parts include the failure modes as defined in the Navy's CASS (Consolidated Automated Support System) Red Team Package[7]. Users can edit the predefined failure modes or add their own catastrophic or parametric failure modes.

Failure modes are simulated by programmatically generating the proper SPICE 3 syntax to describe the failure. Failure modes can be setup for primitive (resistor, transistor, etc.) as well as subcircuit macromodel-based elements. 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 arbi-

Table 1, SPICE 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 Parametric fault	Q1 12 19 24 QN2222 .MODEL QN2222 NPN AF=1 **BF=105** BR=4 CJC=15.2P CJE=29.5P…	Q1 12 19 24 Q1_Fail .MODEL Q1_Fail NPN AF=1 **BF=10** BR=4 CJC=15.2P CJE=29.5P…
Resistor Stuck 2V below Vcc	R1 6 0 1K	R1 6 0 1K Rstuck_6 6_Stuck 6 10.00000 Bstuck_6 6_Stuck 0 V= Vcc- 2
Timed Dependent Inductor Fault	L2 3 0 62U	L2 3 0 62U Rstuck_3 3_Stuck 3 10.00000 Bstuck_3 3_Stuck 0 V= Time > 10n ? 0 : V(3)

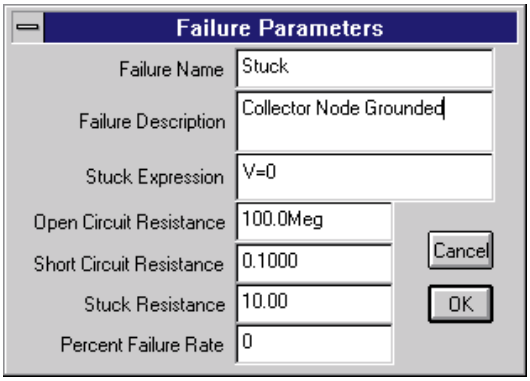


Figure 3, An example of the dialog which is used to define the failure mode characteristics.

trary dependent source which is capable of analog behavioral modeling[8,9]. Behavioral expressions can contain mathematical equations, If-Then-Else directives, or Boolean logic [10]. 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.

It should be noted that the software builds the B element expressions and SPICE models, inserts the required elements, and generates the SPICE netlist automatically. 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 (examples are shown in table 1). The characteristics (open/short/stuck resistance) of each failure mode can be defined by the user (Figure 3).

V. 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 some type of data post-processing program, must be included to extract and record information from each desired test point waveform. For the Test Designer software, the former was chosen and implemented using ICL which is available in SPICE 3 and Nutmeg.

The software uses the IsSpice4 simulation engine to perform the failure analysis. IsSpice4 is an enhanced version of Berkeley SPICE 3 [9] and XSPICE [11,12]. 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 new ICL commands which allow SPICE 3 plots, or sets of vectors (i.e. waveforms) to be scanned and measured with cursors. In contrast to traditional SPICE "dot" statements, ICL

Group Delay Measurement

Cursor Script	
HomeCursors	Reset cursor to waveform endpoints
Vsignal = db(???)	Use dB values
Vphase = PhaseExtend(ph(???))	Use phase values
theDelay = -differentiate(Vphase)/360	Differentiate the phase waveform
theMax = max(Vsignal)	Find the maximum
MoveCursorRight(0,Vsignal,theMax)	Move the left cursor to the maximum
Fmax = GetCursorX(0)	Store the frequency
SetCursor(1,Fmax)	Move the right cursor to the maximum
MoveCursorRight(1,Vsignal,theMax-3)	Move the right cursor to the upper -3dB point
MoveCursorLeft(0,Vsignal,theMax-3)	Move the left cursor to the lower -3dB point
Measurement Script	
groupDelay = mean(theDelay)	Store the measurement

Table 2, Example Wizard generated ICL scripts used to automate measurements of the failure analysis. The ??? fields are replaced with the desired vectors which will be processed.

commands are performed in order, one at a time. This makes ICL scripts perfect for describing test procedures.

A "Wizard" approach is taken in order to alleviate the syntax headaches associated with script development. For example, a Cursor Wizard is employed to position one or more imaginary cursors on a waveform or set of waveforms. Y axis cursors are positioned with respect to a single waveform or vector, while X axis cursors are positioned with respect to an entire set of vectors. A Measurement Script Wizard is used to manipulate the data derived from the cursors in order to produce a single measurement.

A variety of functions are available for setting the cursor positions and for measuring the data in between the cursors. Two example scripts are shown in Table 2. 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.

VI . EXAMPLE

Now that we have defined how a design is setup, we can proceed to show how the software performs the simulation, and discuss the failure diagnostic and test sequencing process. This is best done through an example.

The circuit shown in Figure 4 is a forward converter which uses the Unitrode UC1843 PWM and Magnetics MPP5812 core models from the Intusoft Power Supply Designer's Library. The start-up transient waveform (V(5), top right) is shown, along with a close-up view of the output ripple. Because the circuit uses a full nonlinear switch level PWM IC model and an accurate power Mosfet model, we can examine such detailed

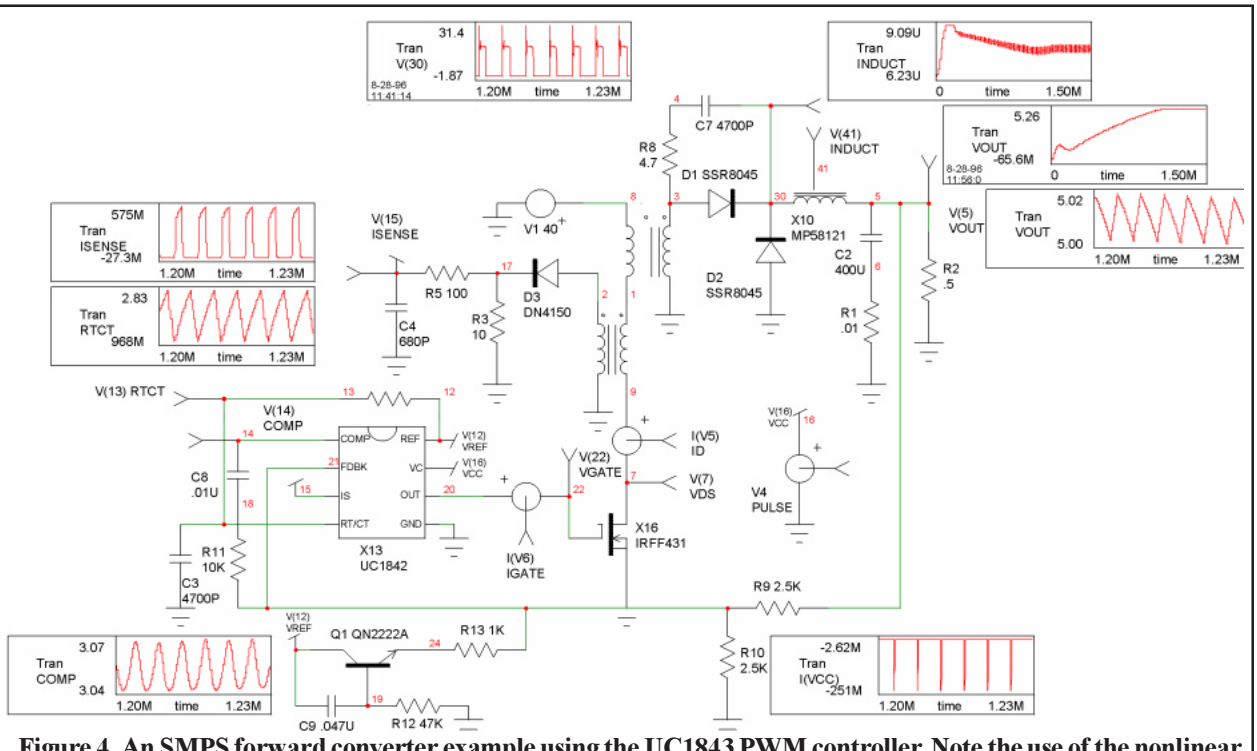


Figure 4, An SMPS forward converter example using the UC1843 PWM controller. Note the use of the nonlinear core (MPP) model. The test point in the center of X10 shows the inductance value as a function of time.

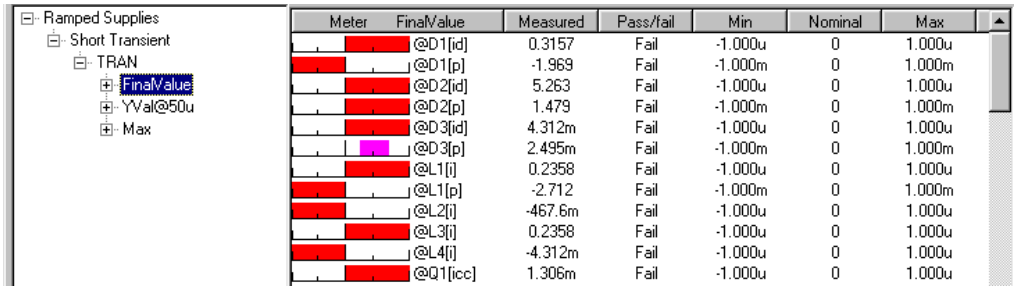
phenomenon as the Mosfet's switching characteristics, operating current into VCC, under voltage lockout threshold, and propagation delay. To simulate the start-up of the circuit, the power supplies V1 and V4 were ramped from zero to their final value, over a 100us interval.

Initially, the SMPS was simulated in full start-up mode for 1.2ms. The simulation runtime was 184.90s. It was decided that a shorter transient run could yield enough useful tests to detect the majority of the faults and be simulated more quickly. For the first failure analysis, a nominal transient analysis of .3ms in length was selected.

The selected measurements were the value at 50us, the maximum value, and the final value at .3ms. The maximum value will be useful for oscillating waveforms, while the final value will be useful for filtered and slow-changing waveforms. The 50us measurement can be used as a comparative measurement of start-up performance. The measurements were determined after looking at the nature of the waveforms and estimating which tests would best be able to detect significant differences when a fault is inserted.

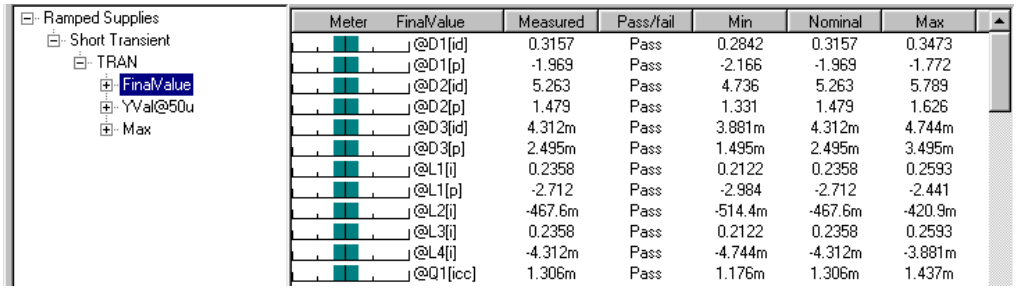
For each of the three measurements, all of the available vectors (circuit test points including voltage current and power dissipation) were recorded. While not all vectors can normally be probed on a circuit card, it is important to gather all of the possible data. Later, when the test sequencing is performed, we can grade the usefulness of each measurement. By measuring all possible test points up front, we eliminate the need to perform subsequent simulations.

The initial results, shown in Figure 5, indicate that all of the tests fail, since no pass-fail tolerances have been applied. Figure 6 shows the results after applying default tolerances of ±100uV for voltages and ±1mA for currents.



Meter	FinalValue	Measured	Pass/fail	Min	Nominal	Max
	@D1[id]	0.3157	Fail	-1.000u	0	1.000u
	@D1[p]	-1.969	Fail	-1.000m	0	1.000m
	@D2[id]	5.263	Fail	-1.000u	0	1.000u
	@D2[p]	1.479	Fail	-1.000m	0	1.000m
	@D3[id]	4.312m	Fail	-1.000u	0	1.000u
	@D3[p]	2.495m	Fail	-1.000m	0	1.000m
	@L1[i]	0.2358	Fail	-1.000u	0	1.000u
	@L1[p]	-2.712	Fail	-1.000m	0	1.000m
	@L2[i]	-467.6m	Fail	-1.000u	0	1.000u
	@L3[i]	0.2358	Fail	-1.000u	0	1.000u
	@L4[i]	-4.312m	Fail	-1.000u	0	1.000u
	@Q1[icc]	1.306m	Fail	-1.000u	0	1.000u

Figure 5, The results of an initial simulation before the default tolerances are applied to the measurements.



Meter	FinalValue	Measured	Pass/fail	Min	Nominal	Max
	@D1[id]	0.3157	Pass	0.2842	0.3157	0.3473
	@D1[p]	-1.969	Pass	-2.166	-1.969	-1.772
	@D2[id]	5.263	Pass	4.736	5.263	5.789
	@D2[p]	1.479	Pass	1.331	1.479	1.626
	@D3[id]	4.312m	Pass	3.881m	4.312m	4.744m
	@D3[p]	2.495m	Pass	1.495m	2.495m	3.495m
	@L1[i]	0.2358	Pass	0.2122	0.2358	0.2593
	@L1[p]	-2.712	Pass	-2.984	-2.712	-2.441
	@L2[i]	-467.6m	Pass	-514.4m	-467.6m	-420.9m
	@L3[i]	0.2358	Pass	0.2122	0.2358	0.2593
	@L4[i]	-4.312m	Pass	-4.744m	-4.312m	-3.881m
	@Q1[icc]	1.306m	Pass	1.176m	1.306m	1.437m

Figure 6, The results of an initial simulation after the default tolerances are applied to the measurements. The Results dialog shows the pass-fail status through the use of a unique histogram indicator (left side).

The results report shows the simulated (Measured column) value, whether the test passed or failed, and shows the minimum, nominal, and maximum values. A special histogram bar icon is used to provide a quick visual indication of the test status.

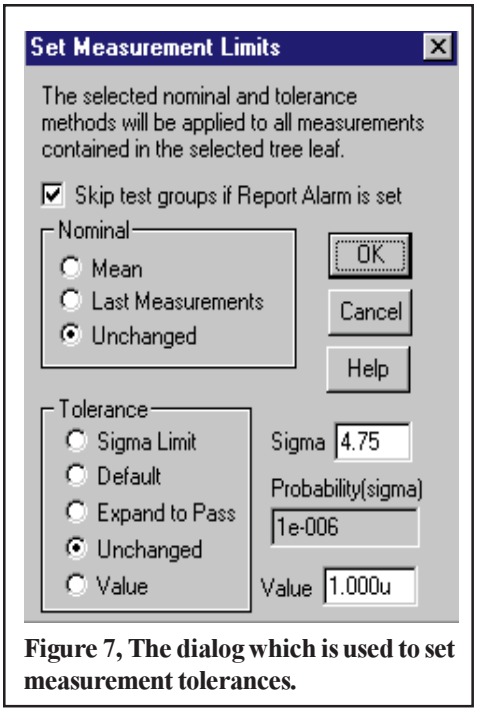


Figure 7, The dialog which is used to set measurement tolerances.

VII. SETTING PASS-FAIL TOLERANCES

A variety of methods are available for setting tolerances including hard limits, Monte Carlo analysis, and a unique "Expand to Pass" method (as shown in Figure 7). Expand to pass moves the min and max tolerance bands outward until the measurement is within the pass band. This allows tolerances to be set through different simulation scenarios such as high and low temperature, or high and low power supply voltage.

Setting limits for the tests is an iterative process of selecting highly reliable tests with regard to detection characteristics, and adjusting the limits on less-than-reliable tests in order to improve their detection characteristics when such tests are required in order to achieve desired isolation metrics.

Test set tolerances and variations can cause measurements to fail. Therefore, a convenient way to account for this is to expand the tolerances by increasing and decreasing the power source values and then using the Expand to Pass feature.

Monte Carlo analysis can also be used to set the measurement tolerances. However, Monte Carlo results tend to yield tolerances that are too tight. A combination of the two methods can also be used. Of course, tolerances can also be set manually on individual measurements or groups of measurements. Figure 8 shows the results dialog for the FinalValue measurement after increasing and decreasing the power supply values by 5% and using the Expand to Pass feature.

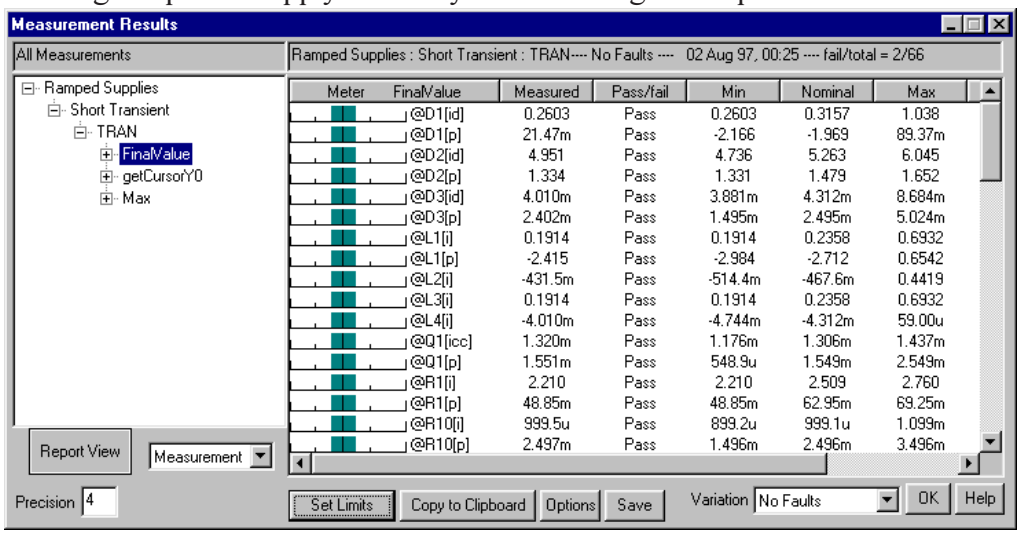


Figure 8, The Results dialog after using the Expand to Pass feature to set tolerances for ± 5% power supply variations.

VIII. FAULT SIMULATION

At this point, a failure analysis is run. Measurements alarms can be set in order to flag failure modes which overstress parts. Once these failure modes are detected, separate tests can be added to find them. They can then be sequenced so that the failure mode test does not destroy functioning parts.

The fault universe for the SMPS consisted of 51 failure modes. Of particular interest were the PWM and power Mosfet faults. The three considered PWM IC failure modes were: Output shorted to VCC, Output shorted to ground, and Output open. The five considered power Mosfet failure modes were: (D=Drain, G=Gate, S=Source) shortedGS, shortedGD, shortedDS, OpenDS, and OpenG.

Failure mode simulation can proceed in one of several ways:

- One fault mode is inserted at a time for 1 test configuration.
- One fault mode is inserted at a time for several test configurations.
- All of the fault modes individually inserted, in succession, for 1 or more test configurations.

In this example, all failure modes are individually simulated for a ramped power supplies configuration, running a short transient analysis. The results are reported in a special dialog which is shown in Figure 9. For each failure mode, the software records the value of every user-defined measurement.

Past work [13,14] implies that simulation runtime is a major inhibitor to this 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 state-of-the-art simulator and computer performance. With the proper application of the these items and control of the simulator options, analysis of a SMPS in the transient domain using fault-by-fault simulation is clearly possible, as the indicated by the following results:

Type of Run	**Analyses**	**Circuit Configuration**	**Time**
Single Simulation	Full Transient	Ramped Supplies	184.90s
Single Simulation	Short Transient	Ramped Supplies	49.18s
Monte Carlo (30 Cases)	Short Transient	Ramped Supplies	23.5minutes
Failure Modes (51)	Short Transient	Ramped Supplies	47minutes

All simulations were performed on a 200Mhz Pentium® processor with 32MB RAM.

IX. RESULTS REPORTING

The results for each failure are reported in a special dialog (shown in Figure 9). A tree list, similar to the Windows 95 Explorer tree, lists each drawing configuration, the simulation setups which are used under that configuration, and the individual analyses for each setup. Folded out from the analysis type are the various measurements that have been defined. There are two other display types; one that shows all of the failure results for each test, and another that shows a histogram of test measurements vs. failure modes, as shown in Figures 10 and 11.

The meter on the left of the report is used as a quick indicator of the measurement's pass-fail status. A long bar on 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 just to the right or left of center 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).

The Variation drop-down contains a list of all of the simulated faults, making it easier to thumb through the results of each fault.

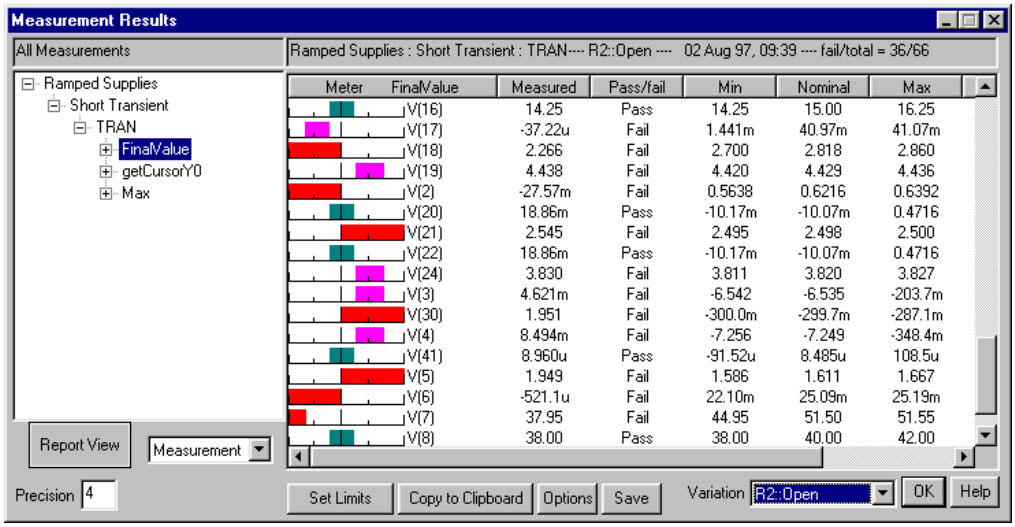


Figure 9, The results dialog after simulating all of the fault modes. The list of measurements for R2:Open are shown.

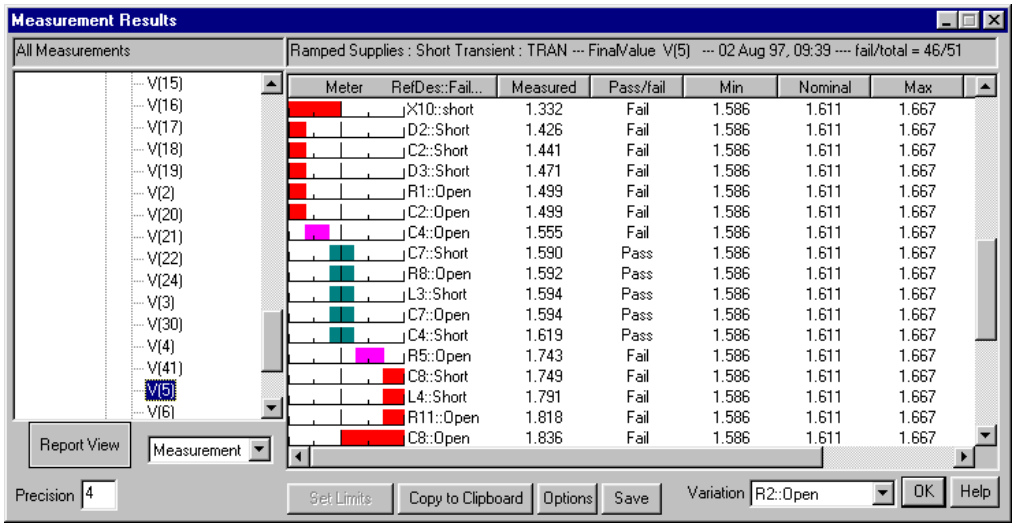


Figure 10, This version shows all of the fault modes results for the final value measurement of the output V(5).

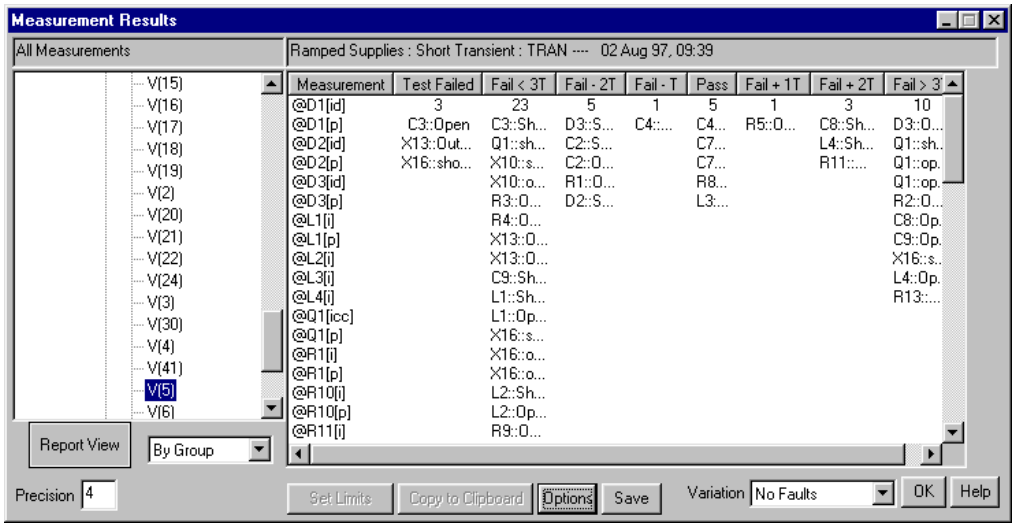


Figure 11, Shows all of the faults for the final value measurement of V(5) using a histogram sorting technique. Failed tests are on the left. The rest of the faults are grouped into bins where T is the pass bandwidth (max-min).

X. ADDING MORE TESTS

Using the test sequencing techniques which are described below, 78% of the faults were detected. X13:Out-toVcc, C3:Open, C2:Open, D1:Short, D1:Open, D2:Short, D2:Open, L2:Open, Q1:ShortCE, R1:Open, and X16:ShortGD were not be detected. Some faults were not be detected from the test data, while other faults caused simulation convergence failures. It is evident that the SMPS diagnostics require several additional test setups in order to completely detect all of the faults. Three other configurations are necessary. They are described below.

Circuit Configuration	Analyses	Faults Detected	Measurement	Description
Dead Circuit Test	AC analysis	C3:Open	Resistance	DMM resistance check
PWM Output Test	Short Transient	X13:Out-to-VCC	Peak-peak	PWM IC output short to Vcc
Reduced Supplies	Short Transient	C2:Open, D1/D2:Short D1/D2:Open, L2:Open Q1:shortCE, R1:Open X16:shortGD	Maximum	Main power supply off PWM supply ramped on slowly

One of the three configurations differed solely in its stimulus and power supply settings. Five schematic layers were created in order to implement these three configurations. The first layer, "Core Circuitry", contains all circuitry for the SMPS. The other layers contained different power supplies for the other configurations, or in the case of the impedance measurement, a DMM instrument. It should be noted that each of the configurations is a stand-alone design. Each has a unique netlist and a unique part list. The common production circuitry is carried across all configurations; this greatly helps minimize transcription errors.

In a second failure analysis pass, the three new configurations were simulated with respective failure modes inserted. The last step, discussed below, involves the sequencing of the tests into a fault tree.

XI. RANKING OF FAILURE STATES

The process of fault detection and fault tree generation takes place in the Fault Tree design dialog (Figure 12) using a novel test sequencing technique.

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 [1,2]. Given an equal probability of occurrence for each fault, the best test is usually 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 [2].

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 on 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 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 fault trees. The software does not attempt to grade tests by difficulty, since this may be very subjective. Instead, 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.

The failure state, (binary, tertiary, histogram or vector), defines the method by which measurements are used to make a test. 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. Here's the way each failure state works to define tests:

1. **Binary:** The test passes if the result is within the test limits, and fails if it is outside of the limits.
2. **Tertiary:** The measurement is divided into 3 states; fail low, pass and fail high. Two tests are generated for each measurement, with outcomes of <pass, fail low> and <pass, fail high>.
3 **Histogram:** Measurements are divided into 7 groups which correspond to the values in the Results dialog's pass-fail meter. There are 6 tests, and each defines one of the failed bands; that is, <fail if in band, else pass>. The nominal band is not needed.
4 **Vector:** Let nFault be the number of faults. Then a vector can be created having nFault members, each containing the measured value for its respective fault. We arbitrarily set a limit about the fault, equal to the nominal tolerance, thus creating an ultimate histogram. Each measurement then has nFault tests, where the fail condition is the case in which the measurement is within the tolerance limits for that vector value.

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.

Vector and histogram-based tests require accurate measurements outside of a circuit's normal operating region. Measurements in this region are inherently less precise because fault models are less precise than nominal models, and faults can place parts that are not failed in unusual or high stress states for which analytic models are less accurate. These facts lead us to conclude that both vector and histogram failure states tend to produce less robust results.

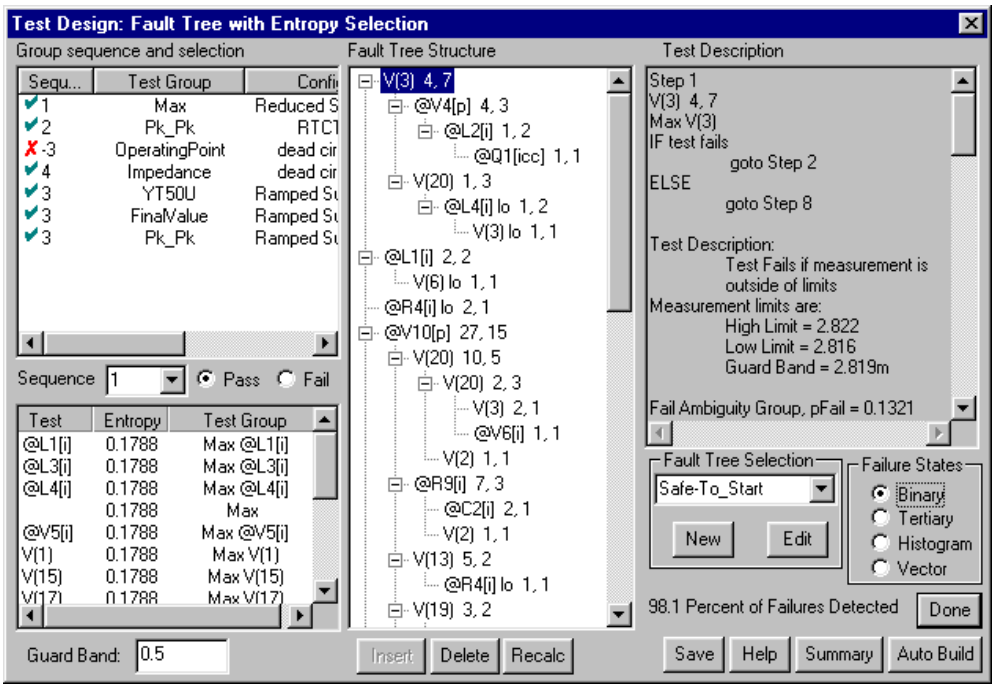


Figure 12, The Fault Tree Design dialog sequences tests into a Fault Tree. The Fault Tree structure for the SMPS example shows the test and the number of faults in the ambiguity groups (pass first, then fail). For instance, L4[I]lo 1, 2 would have 1 fault in the pass ambiguity group and 2 in the fail ambiguity group. The description of each highlighted test is shown to the right.

Measured values can migrate across Failure State boundaries because of component and test tolerances. In order to produce robust tests, the concept of a guard band has been added. 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. You can change the guard band for any Fault Tree step. The guard band width is reported in the Test Description field using the test measurement units.

When measurements are close to test boundaries, UUT tolerances and test set accuracy can move the result across the boundary, yielding a false result. This effect is mitigated by inserting a guard band just outside of the test limits. Setting the test limit to the center of the guard band produces a test that is least likely to give false results.

Referring to the Fault Tree Generation dialog in Figure 12, several groups of tests are available. They are shown in the Group sequence 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 below. In this case, 4 groups were arranged. 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, as shown in Figure 12.

From the failure analysis of our initial configuration, the pass ambiguity group from the last test in the fault tree was used to determine which faults could not detected (22% of the total shown in the "Adding More Tests" section). With all of the tests now available, 100% detection is possible.

Figure 13 displays a sample of some of the report outputs that are produced by the software.

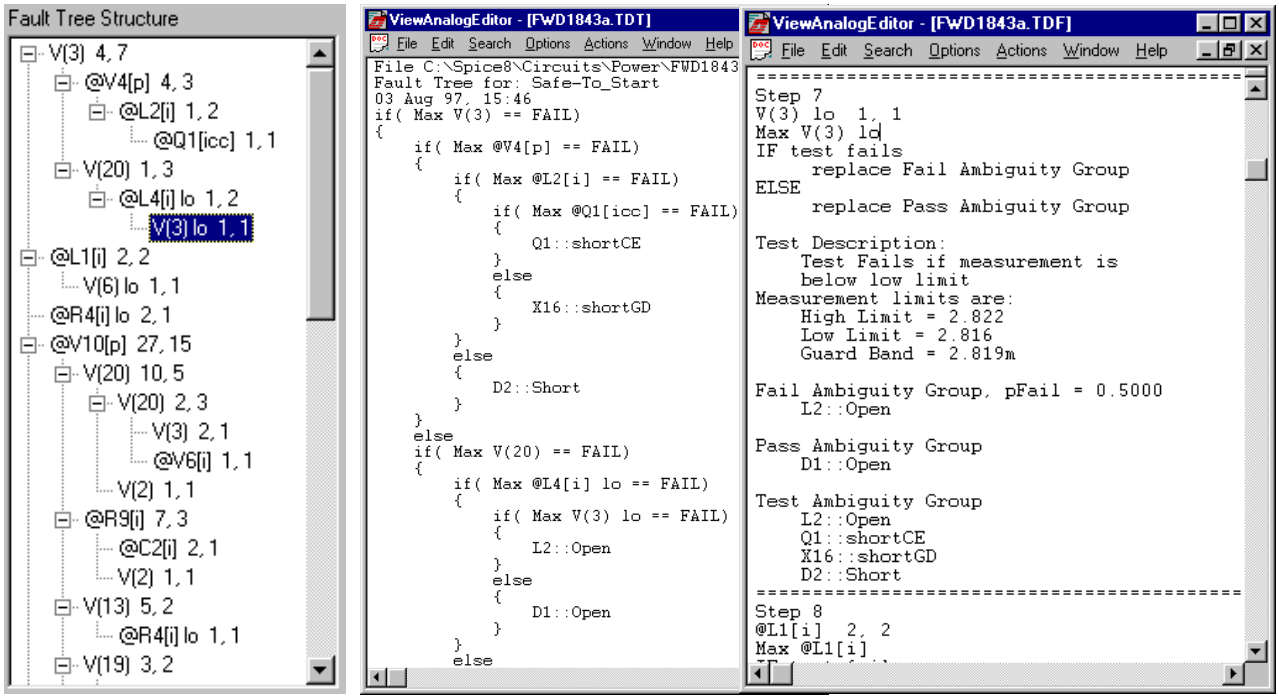


Figure 13, The software produces several types of reports. Shown above is a portion of the fault tree which points to the V(3) test. The matching Test description and hardware independent C-like pseudo-logic code to duplicate the fault tree are shown to the center and right, respectively.

XII. PROBLEMS WITH SENSITIVITY BASED FAILURE ANALYSIS TECHNIQUES

In several recent papers [13,14], 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 author claims that he 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.**

There are two key problems with the sensitivity approach. First, the frequency of part failures near the tolerance limits is but a small fraction of the total failure universe. Second, the linear nature of this prediction makes the implicit assumption that all parameters are related to each output by a first order (straight line) function. It can be shown that even many passive linear circuits fail to meet this criteria!

Using sensitivity analysis to predict circuit performance for extreme values or for component failures only works for certain linear circuits. The basic problem for most analog circuits is that the derivative of a test vector with respect to circuit parameters cannot be guaranteed to predict even the correct sign of the resulting test vector for large parameter changes. For example, consider a simple amplifier which has a gain of zero when its operating point is at either extreme, and a maximum value in between. If the test vector is proportional to gain, then the sensitivity will predict infinite gain for most operating points for one of the extreme fault cases. Even linear circuits can exhibit these characteristics for complex transfer functions; for example, parallel networks can shift complex zeros smoothly from left to right half planes by varying a single component value, yet the output waveform

and its AC phase component will exhibit a discontinuity that can't be predicted using sensitivity analysis. In addition, this technique is generally not valid for circuits which contain switches or switching elements. Mixed signal circuits, by nature, are excluded. Hence, mixed-signal circuits and most power circuits such as PWM ICs can not be analyzed using sensitivity analysis.

In summary, here are a few examples that invalidate sensitivity based techniques:

- Circuits which contain switches. The sensitivity of a switched output with respect to its controlling input is always zero, so no inference is possible. **This applies to all mixed mode circuitry, including most power supply circuitry.**
- Linear circuits which have a non-minimum phase behavior (right half plane zeros). Twin T notch filters are a prime example, as are inverting transistor amplifiers.
- Control systems which have phase shifts of more than 180 degrees. AC sensitivity is based on the adjoint matrix solution at each frequency, so dependence on results at other frequencies is not possible. It is therefore not possible to distinguish between phase planes, making Nyquist stability criteria unavailable.

XIII. CONCLUSIONS

1) The simulation techniques employed here act as a "force multiplier" for development of diagnostics for analog and mixed signal circuits. Over the course of a few hours, it was possible to generate a variety of tests, determine the failure mode detection characteristics of each test, and sequence a subset of these tests into an effective diagnostic fault tree.

2) Simulation is 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.

3) Reasonable simulation times can be achieved by parsing the circuit into separate subcircuits and linking the simulation results through the use of voltage and current stimuli.

4) The software allows the user to study various power supply failure mechanisms.

Accounting for achieved 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 the circuit complexity increases. The preceding techniques provide a significant advantage, by keeping track of detection and isolation of failure modes during generation of the fault tree.

There are several ways to overcome simulation runtime issues. The first is to use a faster computer or many fast computers to perform the runs more quickly and in parallel. A more attractive method for reducing simulation time is to model the switching elements using a linearization technique called state space averaging [2]. Many faults cannot be detected when using this approach, however, the simulation time for the long switching simulations will be reduced by an order of magnitude.

REFERENCES

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

[2] 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.

[3] 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.

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

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

[6] UL508C Standard for Power Conversion Equipment, ISBN 0-7629-0093-8, Copyright © 1996 Underwriters Laboratories Inc., Nov. 1996.

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

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

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

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

[11] 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.

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

[13] 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.

[14] 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.

[15] Steven M. Sandler, SMPS Simulation with SPICE, McGraw-Hill, 1997.