Intusoft Newsletter

Personal Computer Circuit & System Design Tools

Copyright © Intusoft, All Rights Reserved

Issue #52 Feb. 1998

Tel. (310) 833-0710
Fax (310) 833-9658

TEST SYNTHESIS TECHNIQUES

In this article, we lay out a method for test synthesis that is is based upon standards which have been developed for the aerospace industry. Like VHDL in the past, affordable tools are available to migrate this technology to the commercial sector. Benefits of this migration include improved production quality, a reduction in the quantity of spare parts, and dramatically improved time to market. For today's competitive aerospace companies, these techniques are a necessity!

In This Issue

1 **Test Synthesis Techniques**

10 Modeling Gunn Diodes

12 FMCW Radar

16 Why IsSpice is Better!

18 **Worried About Mergers?** *Special Pspice Offer*

18 Year 2000 Compliance

19 Vendor Modeling

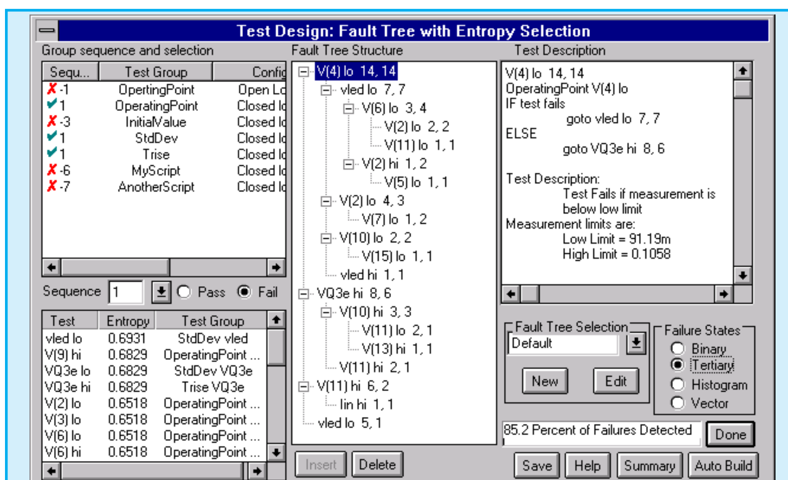


Figure 1, Sample screen taken from Test Designer. Test Designer performs Test Synthesis on analog and mixed-signal circuits. See page 2 to find out how!

How Should A Circuit Be Tested?

Traditionally, the designer specifies test requirements which verify proper functionality of his creation. Frequently, these requirements are just a pass-through of the product specification.

Many contemporary designs are fault tolerant; they may meet their functional specification even if they contain failed or out-of-tolerance parts. Testing only to functional requirements may allow a product to ship with one or more failed components. While these failures may not impair its functionality, as measured by the acceptance test, they increase the odds that the product will fail shortly thereafter.

To remedy this situation, the test engineer or the circuit designer may want to expand the functional test requirements in order to find these component failures. The objective of this "test synthesis" is to define a set of tests that;

a) makes sure that the product works when delivered,
b) rejects a Unit Under Test (UUT) that has failed parts, and
c) enumerates the failures.

Overlaying these objectives is the dual desire not to cause any damage to the UUT through testing, and to perform efficient tests which have robust conclusions. Efficiency minimizes the number of tests that are needed in order to reach a conclusion. Robust testing requires the minimization of false test results.

CASS (Consolidated Automated Support System) Failure Modes

PART	FAILURE MODE
Capacitor	Short, Open
Diode - Rectifying	Short, Open
Diode - Regulating	Short, Open, Change in Vz beyond tolerance
Fuse	Open
Inductor	Short, Open
Logic Device	Outputs Stuck High/Low, Inputs Stuck High/Low, Inputs Open, Power Shorted To Ground
Resistor	Open
Transformer	Shorts between normally non-continuous Windings, Shorts to Grounded Transformer Frame, Open Primary Windings, Open Secondary Windings
Transistor - Bipolar	B-E Short, B-E Open, C-E Short, Collector Open
IGBT/MOSFET	Gate Shorted to other terminal, S-D Short, S-D Open, Gate Open
Linear IC	Outputs Stuck High or at Positive Vcc, Outputs Stuck Low or at Negative Vcc, Outputs Open
Motor - Drive/Servo	Shorted Windings, Open Windings, Binding

Table 1, A partial listing of the failure modes in the U.S. Navy's Consolidated Automated Support System (CASS) specification.

What is a Failure?

A good design can accept part tolerances that are far wider than the tolerances of the individual components. For example, a 2kΩ pull-up resistor could work just as well as the specified 1kΩ resistor. Clearly, we want to accept the out-of-tolerance part in order to take advantage of the increased yield provided by a robust design. Production engineers should be able to substitute parts based on cost and availability.

In the resistor example described above, an open pull-up resistor could actually pass a functional test, and fail in the next assembly when the noise environment increases.

It is reasonable to conclude that we want to detect and reject products which have catastrophic component failures; but we can accept products which have parametric "failures" that do not affect functional performance. Actually, these parametric failures are part of the tolerance distribution of the parts we are using. Monte Carlo analysis will indicate the robustness of the design when we compare the resulting performance predictions with the product specification.

Defining Failure Modes

Component failures are well characterized by a finite number of catastrophic failure modes. For digital circuits, there are 2 failure modes; stuck at logic one and stuck at logic zero. Film and composition resistors are characterized by open circuit failures. Table 1 is the way the US Navy characterizes the catastrophic failures for many common parts. You may define failure modes using common-sense experience, historical data or physical analysis. Depending on your application, you might want to consider additional failures; for example, bridging or interconnect failures in ICs that result in shorts between devices. In a PCB design, you might want to simulate high resistance "finger print - shorts" caused by improper handling of sensitive components.

Abstracting subassembly failure modes to the next assembly is a common error. Consider the case of an IC op-amp. The failures detected and rejected by the op-amp foundry are mostly caused by silicon defects. Once eliminated, these defects will not reappear. At the next assembly, failures are caused by electrical stress from static electricity, operator error in component testing, or environmental stress in manufacturing. This requires defining a new set of failure modes at the next production level. Again, we can usually describe these failure modes using catastrophic events at the device or assembly interface; that is, open, short or stuck for each interface connection.

Unusual Failure Modes are Infrequent

Unusual failure modes get a lot of attention; for example, finding 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 failed parts, then the quality of your product increases 100 fold after these tests are performed. For many products, that increased quality means there will be no undetected failures delivered to the customer.

Detecting Process Faults

Process faults could cause the shift of many parameters simultaneously. When this is a consideration (usually for ICs), 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 fault modes.

What's A Test?

In order to compare one test with another, there must be a precise definition of the test. **A test is defined as the comparison of a measured value with its test limits.** The result of the comparison leads to the conclusion that the UUT either passed the test or failed the test. At the beginning of the test, there is a set of failure modes that have not been tested; these are defined by the Input Ambiguity Group. The test itself is capable of detecting a number of failure modes. These modes are grouped into a set called the Test Ambiguity Group. The pass and fail outcomes then carry a set of failure modes which are the result of logical operations carried out between the Input Ambiguity Group and the Test Ambiguity Group such that:

Fail Ambiguity Group = Input Ambiguity Group AND Test Ambiguity Group
Pass Ambiguity Group = Input Ambiguity Group MINUS Fail Ambiguity Group

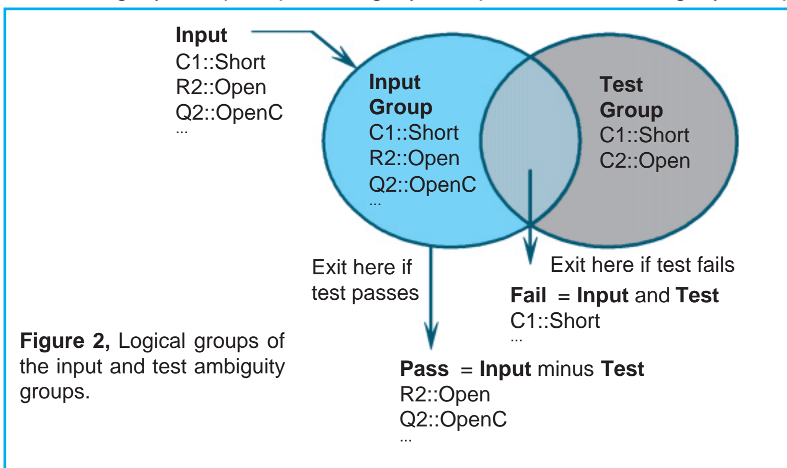


Figure 2, Logical groups of the input and test ambiguity groups.

where AND represents the intersection of the lists, and MINUS removes the elements of one group from the other. Using MINUS here is a convenient way of avoiding the definition of NOT (Input…), since we really aren't interested in a universe that's greater than the union of the Input and Test ambiguity groups. Figure 2 illustrates this logic.

The fail or pass outcomes can then be the input for further tests. If further tests are only connected to the pass output, then a product acceptance test is created. If tests are connected to both the pass and fail outcomes, then a fault tree is created; the fault tree isolates faults for failed products. In either case, the object of each test is to reduce the size of the pass ambiguity group. When the addition of more tests can't further reduce each pass ambiguity group size, the test design is considered to be be completed.

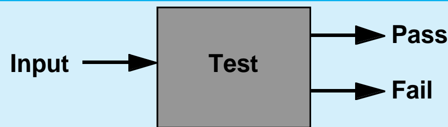
It turns out that characterizing the "no fault" case as a member of the initial ambiguity group, or failure universe, will be useful later on when we decide which test is the best.

Selecting The Best Test

A terminal conclusion is defined as the pass or fail conclusion for which no more tests can be found. Then, if the no fault mode is present, it is the product acceptance test result, with all remaining faults being the ones that are undetectable. Otherwise, the parts with the resulting failure modes are the set that would be replaced in order to repair the UUT.

In general, our goal is to reach the terminal pass/fail conclusions by performing as few tests as possible to reach each conclusion. The general solution to this problem using an exhaustive search technique expands too rapidly to find a solution during the lifetime of our universe… it's an N-P complete problem. Several heuristic approaches are possible, one of which follows.

If we take the idea of failure modes one step further, we can give each failure mode a failure weight that is proportional to the failure rate. To avoid looking up failure rates, we can default these weights to 1.0, and later we can assign a more precise value. The weights will be used to prioritize the search for the best test, and weighting the test that isolates the parts with higher failure rates first. For each test candidate, we can compute the probability of a pass outcome and a fail outcome. From a local point of view, the summation of the pass and fail probabilities must be unity; that is, the UUT either passes or fails a particular test. Borrowing from information theory, we can compute the test entropy as shown in Figure 3:



Let p = probability of a pass outcome
and q = probability of a fail outcome

Then $p + q = 1.0$ and
$p = \Sigma$ Pass weights / Σ Input weights
$q = \Sigma$ Fail weights / Σ Input weights

$$\text{Entropy} = -p \log(p) - q \log(q)$$

Figure 3, Calculation of the test entropy.

The highest entropy test contains the most information. We select the best test as the test which has the highest entropy. For the case when failure weights are defaulted to unity, this method will tend to divide the number of input failures into 2 equal groups. Since "no fault" can only be in the pass group, a high "no fault" weight will steer the tests through the pass leg fastest, making the best product acceptance test. The rationale for a high "no fault" probability is the expectation that most units will pass the production acceptance test; this is a condition of an efficient and profitable business. If, on the other hand, we want to test a product that was broken, we would give the "no fault" probability a lower value. Then the test tree would be different, having a tendency to isolate faults with fewer tests.

Test Sequencing

The definition of the best test did not include the difficulty of setting up the test or performing it; and it didn't include the potential of a failed part to destroy other parts. In addition, the tests were selected independent of the product specification so that we could get to the terminal pass outcome with a tolerance failure. To overcome these problems, tests are sequenced. The sequence priority is:

1. Perform tests that eliminate failure modes that could be destructive in future tests. Frequently, this requires a new test configuration that performs a "safe to start" test.
2. Perform easy tests first, such as DC measurements at room temperature.
3. Perform similar tests in sequence, such as rise time tests which use the same test equipment.

4. Repeat the process for major setup changes, such as measurements under high temperature conditions.

What About Required Tests?

We will discuss robust testing next, however, if we take a hypothetical product specification, we can see that faults will usually show up with wider test limits than the product specification limits. These tighter limits make it possible for tolerance failures to appear as catastrophic failures. If we do fault isolation, the wrong conclusion could be reached. Basically, the required tests need to be placed in as harmless a place as possible, usually at the end of the sequence in which they are found. Functional limits should not be used for fault isolation unless they are also robust.

Robust Tests

Tolerances can cause measurement results to migrate across the test limit boundary. As a result, a fault could be classified as good, or a good part could be classified as a failure. Tolerances include:

- UUT part tolerances
- computer model accuracy
- measurement tolerances
- UUT noise
- test set noise
- fault prediction accuracy

Avoiding false test conclusions requires setting the test limits as far away from the expected results as possible. To compare one test with another, we need to define a measure of test robustness.

A test measurement, for a UUT that is good, has a range of values (called a “tolerance band”) that defines acceptable performance. The measure of test robustness with respect to a failure mode is then the distance between the failed measurement result and the nearest test limit, divided by the tolerance band. We call this value a “guard band”. The test limit can be safely placed in the guard band as long as no other faults have results in this band. Normalizing all measurements using their tolerance band allows us to compare the guard bands of different tests. We can then modify the entropy selection method to reject tests which have small guard bands. Figures 4 and 5 show how this works.

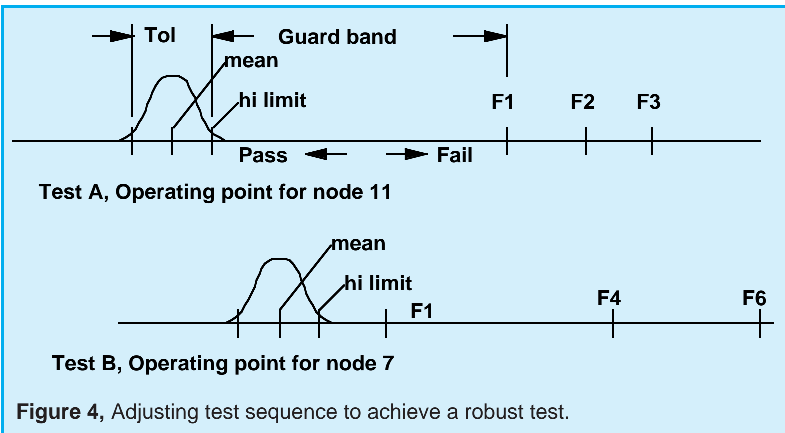


Figure 4, Adjusting test sequence to achieve a robust test.

In this example, we have 2 operating point tests. Failure modes are identified as NoFault, F1, F2, …F6. It is assumed that test A is performed first, and test B is performed on the pass group of test A, as shown in Figure 5. Test A divides the failures into a pass group containing F4,F5,F6 and a fail group containing F1,F2,F3. Connecting test B to the test A pass outcome eliminates F1 from the test B failure input. The guard band for test B extends from the hi limit to F4. If test B were done first, the guard band would be smaller, from the test B hi limit to F1.

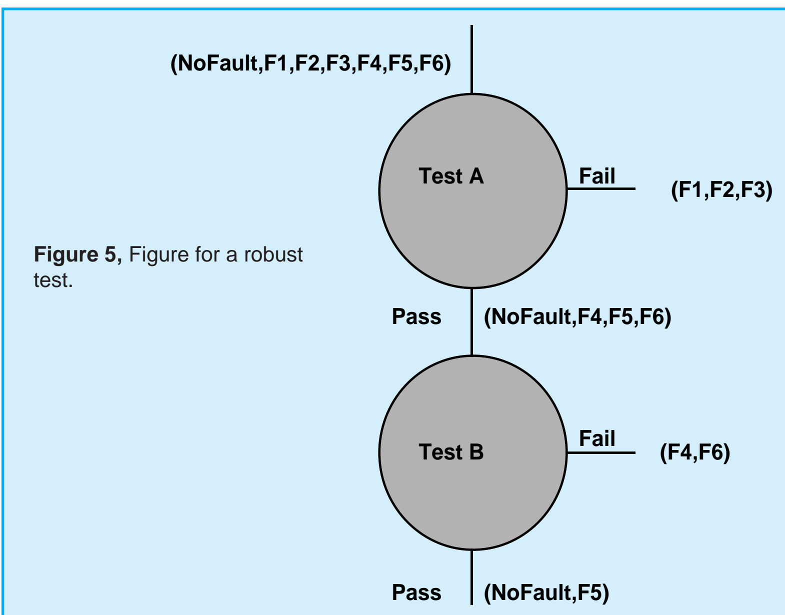


Figure 5, Figure for a robust test.

An incorrect test outcome will invalidate subsequent test decisions. In order to be correct most often, tests with large guard bands should be performed first, since they are less likely to be incorrect. Moreover, tests that were previously rejected may turn out to be excellent tests later in the sequence, as illustrated in the example. Tests with small guard bands simply should not be used.

While a model of the statistical distribution is shown in Figure 4, you should be aware that there usually isn't sufficient information to have a complete knowledge of the statistics of a measurement result. In particular, the mean is frequently offset because of model errors; for example, a circuit that is operating from different power supply voltages than was expected. The statistics of failed measurements are even less certain because the failure modes and the failed circuit states are less accurately predicted. It is necessary, therefore, to increase the tolerance band as much as possible. We avoided stating exactly where in the guard band the measurement limit should be placed; it's a judgment call, depending on how much the tolerance band was widened, and on the quality of the fault predication.

Conclusion

Testing can extend well beyond the traditional functional performance demonstrations of the past. Now you can synthesize tests for analog and mixed-signal circuits and guarantee a fault coverage percentage, just like you do for your digital tests. Moreover, faults can be isolated to facilitate repair or enhance quality management.

Dramatic improvements in EDA software (for example, our Test Designer product - see Figure 1 on page 1) has enabled these comprehensive test development features to be included in any design, and at a reasonable cost. You can visit our web site at www.intusoft.com and download a detailed brochure, application notes, and an evaluation version of Test Designer, or email us at test@intusoft.com to arrange for a personal on-site demonstration.

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

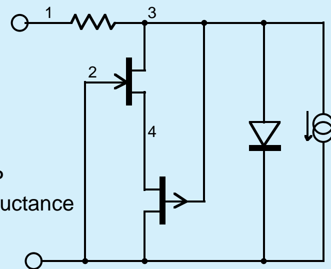
A GUNN DIODE RELAXATION OSCILLATOR

Transferred Electron Devices (TEDs), widely know as Gunn diodes, are gallium arsenide (GaAs) or indium phosphide (InP) devices which are capable of converting direct current (DC) power into radio frequency (RF) power when they are coupled to the appropriate resonator. Typical applications for Gunn diode oscillators include local oscillators, voltage controlled oscillators (VCOs), radar and communication transmitters, Doppler motion detectors, intrusion alarms, police radar detectors, smart munitions, and Automotive Forward Looking Radars (AFLRs).

Gunn Diodes are two-terminal negative-impedance semiconductors which are similiar to tunnel diodes (See Intusoft Newsletter 51, Nov. 1997). They are mainly found in microwave oscillators in the range from ten to several hundred Gigahertz. The negative differential impedance of the Gunn diode may be modeled by a complementary pair of JFETs, as shown in Figure 6.

In Figure 7, an example Gunn relaxation oscillator is shown. Three criteria must be met in order for this circuit to operate properly. First, the DC source resistance must be less than the negative impedance. Second, the load-line must intersect the active characteristic in the negative impedance region. Third, the AC impedance of the DC source must be very high in order to ensure that the bias point becomes astable. The first and last conditions can be met by the

Figure 6 - The subcircuit topology and example netlist for the Gunn diode.

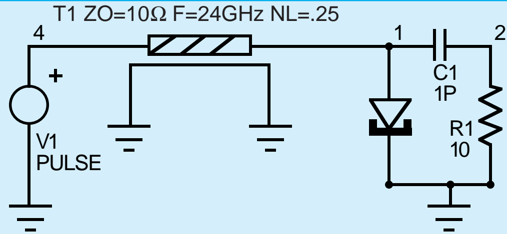



```
.SUBCKT GUNNDIODE 1  2
*Vp=3V Ip=350mA Vv=6.5V
*Iv=50mA Vf=9V Rb=0.5Ω Cj=0.1P
*L1 1 100 0.2nH ; optional lead inductance
R1 1 3 .5
D1 3 2 DIODE
J1 4 2 3 NKANALJFET
J2 4 3 2 PKANALJFET
.MODEL NKANALJFET NJF (VTO=-4V BETA=.07 CGS=.05P)
.MODEL PKANALJFET PJF (VTO=-4V BETA=.07 CGS=.05P)
.MODEL DIODE D (RS=6 N=9)
.ENDS
```

Figure 7 - Gunn relaxation oscillator.

Transient Analysis
.Tran .001n 1n 0 .001n

DC Analysis
.DC V1 0V 12V 0.05V



transforming properties of a quarter-wavelength transmission line. Oscillation occurs at that repetition frequency (see Figure 8) due to the fact that the zero impedance of the DC source appears infinitely high at the gunn diode.

[1] Litton Solid State Division Applications Notes InP and GaAs Gunn Diode Devices, June, 1997.

Figure 8 - The DC characteristics (top) and astable relaxation oscillator response (bottom) using the Gunn diode.

FMCW-DISTANCE RADAR

by Karl Heinz Muller

For more than 40 years, Radar Altimeters have been used in avionics to measure the altitude of airplanes over ground. Advanced developments of Gallium Arsenide millimeter-wave chips have lead to a considerable reduction in costs, thereby allowing this technique to find more and more acceptance with earthbound traffic control systems. After the invention of the air bag and anti-blocking systems, the automotive radar is poised to become the third significant part of this genre of safety equipment.

A Radar Sensor for Advanced Cruise Control

The ideal cruise control would be one that could measure the distance between vehicles and adjust it, depending on the general traffic speed. In order to accomplish this, a self-contained sensor must be mounted on the vehicle.

With a vehicle-mounted radar, it is possible to:

- Measure the position and speed of all objects in front
- Ignore all irrelevant objects (e.g. crash barriers, approaching traffic)
- Distinguish between vehicles in different lanes
- Lock-on to the closest vehicle in the same lane
- Continuously output Range, Speed and Angle for the locked-on vehicle

Frequency-Modulated-Continuous-Wave (FMCW) radar has many applications. In FMCW applications, the distance measurement is derived by a transmitter/receiver combination that is continuously and linearly swept from a low frequency to a high frequency. FMCW radar is based upon on the doppler effect. For example, if the lower and upper frequencies are 2000MHz and 5000MHz, and a 2000MHz signal is transmitted and hits an object, a reflection is received from that object to the antenna. By the time the signal reaches the antenna, the transmitted signal would have been, for example, 2005MHz. The returning signal at 2000MHz and the transmitted signal at 2005MHz are fed into the mixer, yielding an intermediate frequency of 5MHz. The intermediate frequency is directly proportional to the distance to the reflecting surface.

The FMCW-distance radar (see Figure 10) is intended for collision avoidance and cruise control on heavily loaded highways. Due to the high speed of electromagnetic propagation, an unmodulated high frequency carrier alone is not suited to

```
.SUBCKT FMCW_OSCILLATOR  1  2
* Carrier_Freq = 200MEGHZ
* Unilateral_Freq_Deviation = 50MEGHZ
* Modulation_Freq fm = 1KHZ
* Modulation_Index = 50MEGHZ/1KHZ =50000
V1 1 0 SFFM 0 10V 200MEGHZ 50000 1KHZ
R1 1 2 50
.ENDS
***
.SUBCKT ANTENNA_TARGET_AND_BACK  1
* Target_Range=150m <==> Travel_Delay=2 * 0.5US
* Target_Range=15m <==> Travel_Delay=2 * 0.05US
T_RANGE 1 0 2 0 Z0=50 TD=0.5US
R_TARGET 2 0 52
.ENDS
***
.SUBCKT MIXER  1  2  3
*                    LO RF IF
* Average IF  fd=4 * 50MEGHZ * 1KHZ * 1US = 200KHZ
* Average IF  fd=4 * 50MEGHZ * 1KHZ * 0.1US = 20KHZ
E_MULTIPLIER 4 0 POLY(2) 1 0 2 0 0 0 0 0 1
R1 1 0 50
R2 2 0 50
L1 4 3 15.9U
C1 3 0 15.9P
R3 3 0 1K
.ENDS
```

Figure 9 - Models for the FMCW Radar system including a mixer, antenna target and return, and oscillator.

detect a travel delay of a received waveform caused by reflections from a distant target. With pulse modulated carriers, for example, the leading edges of the leaving and returning pulses can be utilized for this purpose. As an alternative, FMCW systems use the instantaneous carrier frequency shift between the transmitted and received signal to measure the delay, td=2r/c, which is proportional to the distance, r. This is accomplished via a mixer which operates as a multiplier for both signals, by generating sum and difference frequencies. While the sum is suppressed by a low pass filter, the difference appearing at the IF port is used for further signal processing.

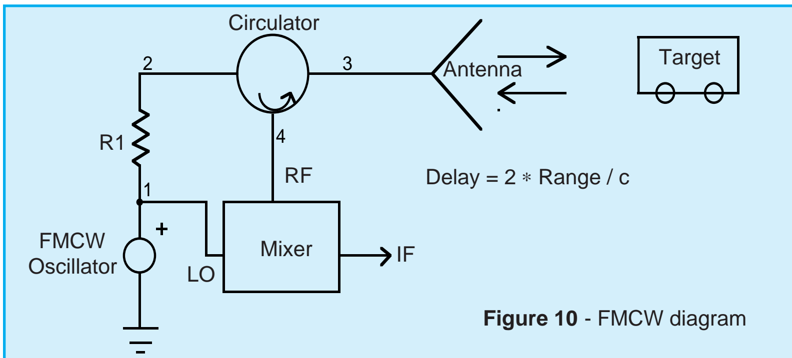
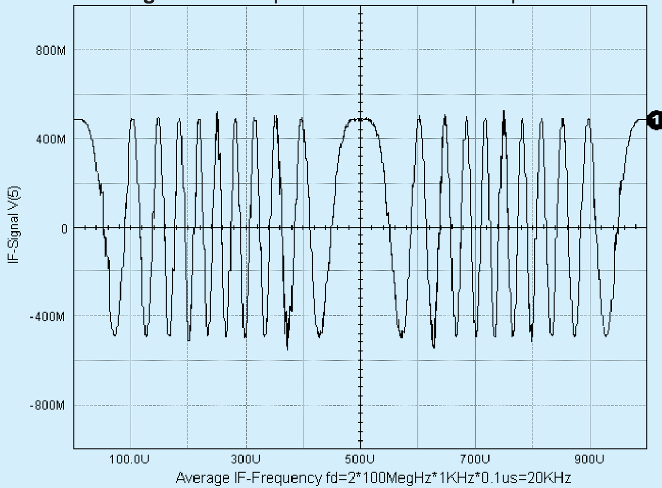


Figure 10 - FMCW diagram

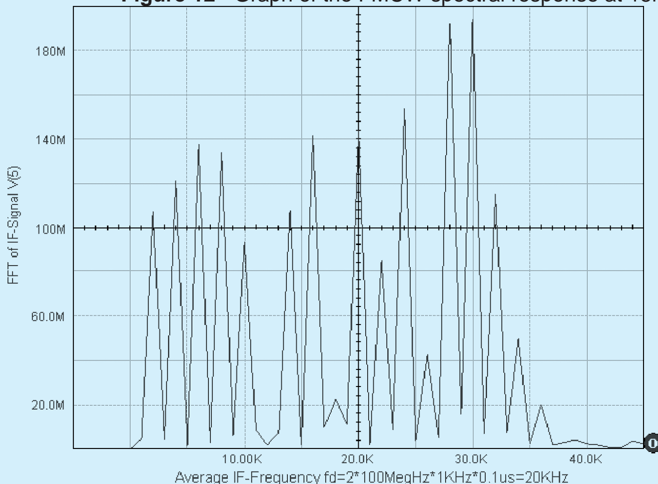
Figure 11 - Graph of the FMCW time response at 15m.



A circulator can be used to separate the transmitting and receiving paths. Therefore, only a single antenna is needed for both channels. Modeling of the distance and reflecting cross section of the target is realized by a slightly mismatched bidirectional delay line. Depending on the standing wave ratio, or reflection coefficient, only a small fraction of the transmitted power is reflected and appears at the RF port of the mixer. As shown by the simulation (Figures 11 and 13), the intermediate frequency is proportional to the distance of the target.

The average frequency may be calculated by the equation:

Figure 12 - Graph of the FMCW spectral response at 15m.



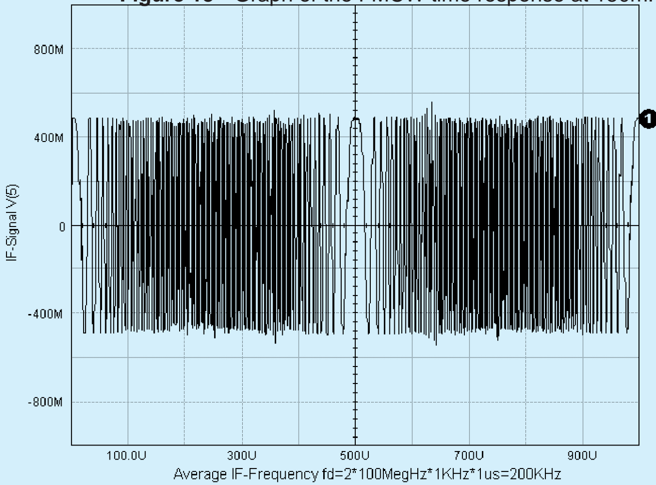


Figure 13 - Graph of the FMCW time response at 150m.

$$f_d = 4 \cdot f_m \cdot \Delta f \cdot t_d$$

The result depends upon the modulation frequency, the unilateral carrier frequency deviation, and the total travel delay time. A more detailed Fourier analysis (see Figures 12 and 14) reveals that the total IF spectrum consists of integer multiples of the modulation frequency whose amplitudes follow the Bessel function. The averaging process may simply be done by a counter that is gated by the modulation period (1/fm). By selecting proper scaling, the counter then displays the true distance of the target.

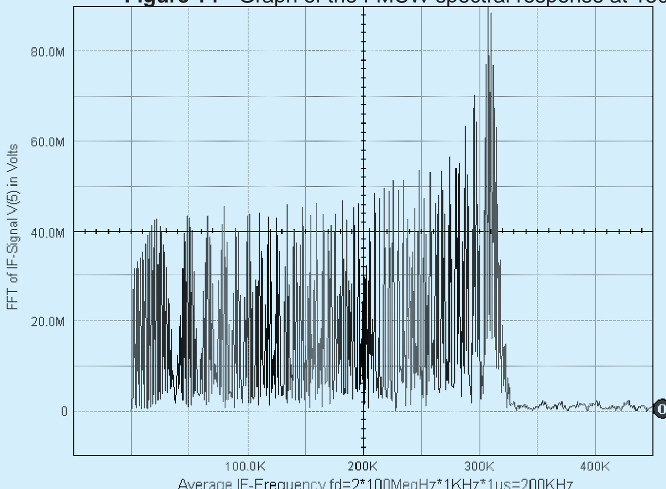


Figure 14 - Graph of the FMCW spectral response at 150m.

IsSpice4 - State-of-the-Art SPICE

IsSpice4 provides a quantum leap in performance over other analog and mixed mode simulators. It is the first commercially available version of SPICE based on Berkeley SPICE 3F and Georgia Institute of Technology's XSPICE.

IsSpice4 allows you to explore circuit performance by interactively running different analyses and sweeping any circuit variable. With the ability to simulate electrical, sampled-data, mechanical, physical, thermal, and other systems, **IsSpice4 is the ONLY true native mixed mode SPICE 3 based simulator.** The advanced features of IsSpice4 allow all types of applications to be simulated: switch mode power supplies, mixed signal ASICs, RF communication systems, interconnects, control systems, and mixed mechanical/physical systems.

Intusoft has spent hundreds of man-hours improving SPICE. And although IsSpice4 is based on SPICE 3, Intusoft has greatly enhanced the program over and above the public domain version; adding an interactive interface, providing superior analysis and model support, and improving the convergence algorithms - all for a price no one can match. IsSpice4 is simply the best and most affordable SPICE program on the market today. Just take a look at some of its features:

State-of-the-Art Operation

- Native mixed mode simulation - IsSpice4 includes an event driven simulator that supports mixed analog, digital and DSP circuits
- Interactive Operation - IsSpice4 operates interactively, and frees you from the restrictive batch style of older SPICE simulators
- Interactive Command Language - Comprehensive set of functions for batch style control of the simulator
- ***NEW VISUAL BASIC SCRIPTING*** - Drive IsSpice4 using VB scripts from popular programs like Excel
- ***NEW OLE INTERFACE*** - Develop your own OLE/ActiveX interfaces

Built-in Models

- Elements: Resistors, Capacitors, Inductors, Coupled Inductors, Transmission Lines, Diodes, BJTs, JFETs, MOSFETs (Level 1-8), GaAs Mesfets, Switches, and Boolean logic expressions
- Digital and AHDL Models: Digital primitives, State Machine, Frequency Divider, RAM, Sampled-Data Filters, Nonlinear VCOs, Laplace Equations
- Behavioral Modeling: In-line Equations, Table models, If-Then-Else

Advanced Models

- HDL Models and C Subroutines; Create models based on a powerful nonproprietary HDL using C
- Support for nonelectrical applications and top-down system design
- Three types of digital/mixed mode modeling
- Lossy (distributed) transmission lines with frequency dependent losses
- MOS: BSIM1, BSIM2, BSIM3 version 2 and 3.1 and SOI MOSFET models
- MESFET: Statz, Curtis-Enttenburg, Parker-Skellern, and HEMT models

Analysis Support

- AC, DC, transient, noise, Fourier, distortion, DC/AC sensitivity, Pole-Zero analyses, and Temperature variations on individual elements
- Monte Carlo Analysis, Circuit Optimization/Performance Analysis
- ***NEW DESIGN VALIDATOR***™ for automatic design verification
- ***NEW TEST DESIGNER***™ Fault Analysis and Software Test Set Design

Additional Interactive, AHDL & Mixed Mode Features

- Real-time Display of voltages, currents and power dissipation
- Simulation Scripts: a robust scripting language that allows simulation breakpoints and loops of different analyses to be run as a test procedure
- Interactively run analyses without having to edit the netlist or restart the simulator, add, delete, or rescale waveforms on the real-time display
- Digital Simulation: IsSpice4 includes a 12 state digital logic simulator and models with timing information
- Sweep parameters one at a time or in groups with great ease
- Start, stop, pause, change, or resume any analysis on demand
- Use C code subroutines & AHDL models based on XSPICE

Convergence and Speed Improvements

- Automatic Gmin stepping/Source stepping algorithms
- New Pseudo-Transient algorithm
- Improved Predictor-Corrector, Latency, and Bypass algorithms
- Improved program defaults
- Special Circuit Debugging Options
- Full Gear Integration option

Compatibility

- True OLE Integration with popular schematic entry programs
- Pspice® parameter passing syntax compatibility

DID YOU KNOW INTUSOFT WAS THE FIRST?

The following is a list of capabilities that Intusoft introduced to the analog simulation world.

- **SPICE 2 Models for:** IGBTs, fuses, lasers, vacuum tubes, generic template models, dual gate Mosfets, SC filters, neural networks, digital gates, RF beads, IBIS buffers, saturable cores, and PWMs (using the state space approach)

- **Products/Features:** Monte Carlo analysis for PC-based SPICE, integrated schematic entry dedicated to SPICE, SPICE 2 compatible model generation software, 32-bit version of SPICE for DOS, support for all Macintosh platforms, parameter passing

A SPICE FOR EVERYONE!

Affordability is the hallmark of Intusoft's products. With our powerful ICAP/4Rx, ICAP/4Windows, and ICAP/4Macintosh software, all affordably priced, there's an IsSpice that's just right for you.

WORRIED ABOUT MERGERS??

Will your simulation vendor ever supply these features?

- ✔ Interactive Simulation
- ✔ Configurable Schematics
- ✔ Automated Measurements and Design Verification
- ✔ Automatic Failure Analysis and Test Synthesis
- ✔ Automatic Stress Reporting and Alarms

Is your specialty supported?

Analog - Mixed-Signal - Power - RF - System - Test - FMEA

It's time to turn to Intusoft for proven technology from a stable company that you can count on. Intusoft has a reputation for placing state-of-the-art technology on your desktop and providing you with the support and training needed to make your job easier, faster and more productive. Unlike other companies, we don't force you into a single solution. You can integrate our simulation suite with 3rd party schematics, or use our advanced configurable schematic tool to drive your design.

For A Limited Time Offer

PSpice Users can turn to Intusoft now; by simply purchasing a maintenance contract and a training class, you will receive a full ICAP/4 package, along with a 30% discount on most other Intusoft products. Call Intusoft for details, or email help@intusoft.com.

YEAR 2000 COMPLIANCE

Much ado has been made of the year 2000 calendar problem for computers. For the most part, the problem exists for legacy software that compares dates using the last 2 digits of the calendar year; Intusoft software doesn't do this. Intusoft uses dates only for printing on reports and checking file status.

Sample IsSpice4 banner

******* Fri Jan 02 08:59:50 ******* IsSpice4 ver. 7.6.4 ******* 8/15/98

We assume the user will know whether his simulation took place in the year 1900 or the year 2000, so unless there is a strong user sentiment to change the year to 4 digits, we plan no change in this output format. After all, if people can recognize the year, so should computers. The time application interface that we use counts time in seconds, beginning in the year 1970. It uses a signed long variable to hold the value - it can hold up to 68.1 years in seconds; the end of time will occur in the year 2038, at least for this method of counting. Long before this happens, we expect to switch over to 64 bit systems, so the problem will simply vanish as the technology matures.

What vendor has the most model types? What vendor has published 3 books on SPICE and modeling? What vendor has the only Windows based AHDL language? What vendor provides their own SPICE training? What vendor has the most behavioral modeling features? What vendor has a FREE modeling service for its customers?

No Contest - The only answer is Intusoft!

No SPICE Vendor can match the variety and quality of Intusoft.

- **Total Library Size Now Over 13,000 parts, hundreds of model types**
- **User Defined C Code Models for analog, digital, array, and sampled-data elements**
- **AHDL Models using a Windows C based language**
- **Power Library (500+ models from Unitrode, Cherry, Linear Tech, Siliconix)**
- **RF Device Library (600+ models from H-P, NEC, Philips, Motorola)**
- **Vendor supplied IC/Op-Amp Models (1600+ models from Analog Devices, TI, APEX, etc.)**